A Formal Account of the Open Provenance Model

Natalia Kwasnikowska, Hasselt University and transnational University of Limburg
Luc Moreau, University of Southampton
Jan Van den Bussche, Hasselt University and transnational University of Limburg

On the Web, where resources such as documents and data are published, shared, transformed, and republished, provenance is a crucial piece of metadata that would allow users to place their trust in the resources they access. The Open Provenance Model (OPM) is a community data model for provenance that is designed to facilitate the meaningful interchange of provenance information between systems. Underpinning OPM is a notion of directed graph, where nodes represent data products and processes involved in past computations, and edges represent dependencies between them; it is complemented by graphical inference rules allowing new dependencies to be derived. Until now, however, the OPM model was a purely syntactical endeavor. The present paper extends OPM graphs with an explicit distinction between precise and imprecise edges. Then a formal semantics for the thus enriched OPM graphs is proposed, by viewing OPM graphs as temporal theories on the temporal events represented in the graph. The original OPM inference rules are scrutinized in view of the semantics and found to be sound but incomplete. An extended set of graphical rules is provided and proved to be complete for inference. The paper concludes with applications of the formal semantics to inferencing in OPM graphs, operators on OPM graphs, and a formal notion of refinement among OPM graphs.

Categories and Subject Descriptors: H.1.m [**Models and Principles**]: Miscellaneous

General Terms: Languages, Standardization, Theory, Verification

Additional Key Words and Phrases: Provenance, Temporal reasoning, World Wide Web

1. INTRODUCTION

In the context of the Web, *provenance* is information about entities, activities, and people involved in producing a resource (a piece of data, or any other thing) [Moreau et al. 2013]. This information can be used to form assessments about the quality, reliability, or trustworthiness of the resource. On the Web, data flows across multiple systems, implemented using different technologies, and potentially hosted by different institutions. Hence, tracking the provenance of data in this context is particularly challenging, since it involves understanding flows of information in these different systems. There is a rich literature on provenance tracking mechanisms in e-science [Simmhan et al. 2005], in databases [Buneman et al. 2008; Cheney et al. 2009], and in the Web [Moreau 2010b]. In the present paper, however, we are concerned with providing logical foundations for provenance information that has already been collected.

Following a strong community momentum [Gil et al. 2010], there now exists a standard provenance interchange format, given by the PROV-DM W3C Recommendation [Moreau et al. 2013]. With PROV-DM in place, provenance information will increasingly be shared, copied, integrated, and compared. In order to perform these activities in a meaningful manner, it is important that we have a formal semantics for provenance information. Such a formal semantics will give us a formal criterion for when two provenance instances are equivalent, when one instance subsumes another one, or when one is refinement of another one. In this paper, we present a proposal for such a semantics.

We take a logic-based approach and view a provenance instance as a logical theory. Our approach is temporal: the axioms of the theory state inequalities on temporal variables. These variables represent the timepoints of creation of artifacts, of beginning and ending of processes, and of usage of artifacts by processes. A *model* of a provenance instance is an assignment of concrete timepoints to variables that satisfies the axioms of the theory. We can then simply say that two provenance instances are equivalent if they have the same

models; or we can define appropriate notions of refinement or subsumption in terms of logical implications among theories.

We develop our work in the context of the Open Provenance Model (OPM) [Moreau et al. 2011], which has been very popular as a precursor to the PROV-DM standard [Missier and Goble 2011; Miles 2011; Groth and Moreau 2011; Freitas et al. 2011; Lim et al. 2011; Kwasnikowska and Van den Bussche 2008]. OPM is more lightweight than PROV-DM and thus a more tractable data model for formalization. At the same time, the essential elements of provenance instances, namely entities and activities, and the relationships that connect entities or activities with other entities or activities, are present in OPM as well as in PROV-DM; only the terminology is a little bit different. As a matter of fact, our work begins with a critique of OPM, enhancing it with a number of improvements, which have also been adopted by PROV-DM.

In OPM, provenance instances are represented as graph structures. In the setting of our improved OPM model, we make the following contributions.

(1) Our main result (Theorem 4.7) provides a direct, clear and concise characterization of the logical consequences of an OPM graph, in terms of a comprehensive list of patterns that may be matched against the graph.
(2) Our characterization semantically justifies the inference rules that are part of OPM; one of the main corollaries of our main result is a completeness theorem for these inference rules (Corollary 4.13).
(3) We clarify the effect of cycles in OPM graphs by characterizing the equalities on temporal variables that such cycles entail (Proposition 5.10).
(4) Based on our formal semantics, we propose a notion of *refinement* among OPM graphs (Section 6.3).
(5) We define a powerful operation on OPM graphs, called "proper and legal merge-renaming", and show that this operation always results in a refinement (Theorem 6.9).

Since an initial draft of this paper was circulated [Kwasnikowska et al. 2010], the temporal inequalities we propose in this paper have become part of the W3C PROV-CONSTRAINTS Recommendation [Cheney et al. 2013]. Moreover, our notions of interpretation and model have been adopted in the W3C PROV-SEM Working Draft [Cheney 2013]. Thus, many ideas of the present work have percolated into the PROV standard, so that the above list of contributions apply to PROV as well. A detailed discussion of how our results can be applied to PROV will be given in Section 8.

The formalism introduced in this work is amenable to effective implementation, as has been demonstrated by the implementation of several PROV validators. An online PROV validator developed in Java [Moreau et al. 2014] is available at provenance.ecs.soton.ac.uk. The validator prov-check developed by Groth[1] retrieves the patterns from our Theorem 1 by means of SPARQL queries against an RDF representation of the PROV instance. Cheney and Cresswell[2] have made an implementation in Prolog. Moreover, based on an earlier draft of this paper [Kwasnikowska et al. 2010], Dey et al. [Dey et al. 2013] have implemented tools (using Datalog) to generate temporal models for OPM graphs as formally defined in this paper.

To conclude, we would like to stress that a temporal approach is only one possible approach to give a semantics to OPM graphs. Still, it is an approach that follows naturally from the informal explanations of dependencies in OPM graphs given in the reference specification. Also, temporal ordering of events is a fundamental approach already known from the distributed systems literature [Lamport 1978; Mattern 1989; Tel 1994]. It remains an interesting topic for future research to explore alternative approaches, e.g., using Halpern

[1] https://github.com/pgroth/prov-check
[2] checker.pl: https://github.com/jamescheney/prov-constraints

and Pearl's notions of actual cause and explanation [Halpern and Pearl 2005] as suggested by Cheney [Cheney 2010]. We will discuss Cheney's paper together with other related work.

This paper is organized as follows. We begin by reviewing the basic features of OPM, showing the need for a formal semantics, in Section 2. Section 3 then formally defines OPM graphs and their temporal interpretation. In Section 4, the notion of OPM inference, which allows new edges to be inferred, is defined and characterized with respect to the temporal semantics. Given that OPM graphs are meant to be exchanged and manipulated to address provenance use cases, we formalize common operations over OPM graphs in Section 5. From the outset, it was envisaged that relations between OPM graphs, such as refinement, would be of value to reasoners; however, no precise definition of refinement has been proposed so far. This problem is tackled in Section 6, where a purely semantic definition of refinement is proposed, based on the temporal semantics. The notion of account is formalized in Section 7. Related work specific to OPM is discussed in Section 8; that section also presents an explicit mapping of our temporal axioms into PROV-CONSTRAINTS. Conclusions and topics for further research are presented in Section 9.

2. OPM: GRAPHS AND INFERENCE RULES

The OPM reference specification defines an OPM graph as a directed, edge-labeled graph. Nodes can be of two types: artifacts[3] and processes. Accordingly, there are four types of edges: generated-by, used, derived-from, and informed-by, depending on the type of their source and destination:

type of source	type of destination	type of edge
artifact	process	generated-by
process	artifact	used
artifact	artifact	derived-from
process	process	informed-by

Generated-by and used-edges are labeled with so-called roles, which can be likened to field names in records, or to parameter names in procedures.[4]

Example 2.1. Let us consider an e-shop over the Web, making a variety of e-material available to its customers, but also acting as a market place for products sold by third parties. Figure 1 shows an OPM graph with three processes 'Take Order', 'Deliver', and 'Third Party Process', and seven artifacts 'billing address', 'order', 'invoice info', 'delivery request', 'invoice', 'e-book', and 'toy'. The graph contains provenance information about an e-book and a toy that have been bought from an e-shop. As soon as the order was taken, a delivery request is generated. The billing address is provided separately; once provided, the information necessary to make an invoice is sent to the delivery department as well.[5] Upon receiving the delivery request and matching invoice information, the delivery department generates a paper invoice which is sent to the customer; furthermore, an access code for the e-book is made available through the Web transaction. Toys are delivered by a third party. Here the information is less detailed, simply because the observer that generated it does not have access to the third party system; all that is given is that the third party was

[3]In the context of the Web, an artifact would be regarded as a Web resource [Jacobs and Walsh 2004] in a given state. A process is an execution of a program, whether it is a service on the Web or a client-side script, running in a browser.

[4]In the reference specification, informed-by is called 'triggered-by' instead. The reference specification also speaks of *accounts*, which we defer to Section 7. It further introduces *agents* and their incident controlled-by edges, which are the sole feature of OPM which we ignore in the present work, since OPM does not allow provenance of agents to be expressed, and therefore agents have no bearing on the temporal interpretation of graphs.

[5]The example is slightly contrived in order to allow us to illustrate some subtle points of the semantics of OPM graphs later on.

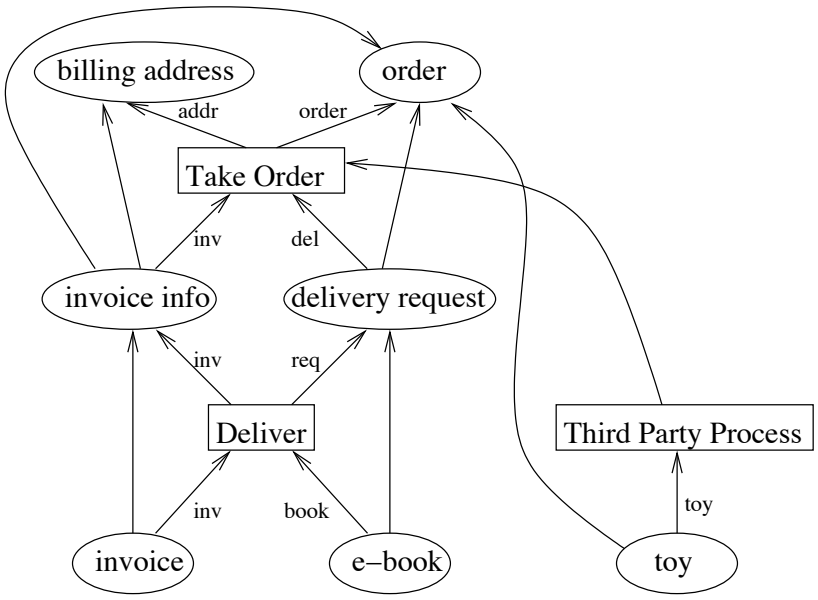

Fig. 1. OPM graph for e-shop order.

influenced by the taking of the original order, and that the toy derives from the same order. Note that there is no derived-from edge from the delivery request to the billing address, as the former may have already been generated before the latter was provided. □

Formally, we have the following.

Definition 2.2 (*OPM graph, first definition*). An OPM graph is a structure

$$(Art, Proc, Roles, GeneratedBy!, Used!, DerivedFrom, InformedBy)$$

where

— Art and $Proc$ are two disjoint finite sets of elements called *artifacts* and *processes*, respectively;
— $Roles$ is a finite set of elements called *roles*;
— $GeneratedBy! \subseteq Art \times Roles \times Proc$;
— $Used! \subseteq Proc \times Roles \times Art$;
— $DerivedFrom \subseteq Art \times Art$;
— $InformedBy \subseteq Proc \times Proc$.

Artifacts and processes are collectively referred to as the *nodes* of an OPM graph. Nodes are identifiers, serving as local references; the actual value of the identifiers is not important as all the information contained in an OPM graph is given by its structure. Nevertheless, when the node identifiers refer to actual objects on the Web, or when different parties want to compare of merge different OPM graphs, some care has to be taken in agreeing on the chosen identifiers. This situation is very similar to URIs in RDF and Linked Data on the Web.

The elements of $GeneratedBy! \cup Used!$ are called *precise edges*, and the elements of $DerivedFrom \cup InformedBy$ are called *imprecise edges*; all together they are called edges. Precise edges are of the form (x, r, y) and are additionally denoted as $x \xrightarrow{r} y$, or, when it is not important to know r, as $x \xrightarrow{!} y$. Imprecise edges (x, y) are denoted simply as $x \to y$.

Note how the above definition allows for multiple precise used-edges between a same process-artifact pair, with multiple roles. They would indicate that during its lifetime a process used a same artifact several times, with different roles.

Remark 2.3. We do not specify that roles are disjoint from artifacts and processes because we do not need that assumption. This is not to say, however, that we attach any importance to an artifact or process that may appear as a role. If an identifier of a node also appears as a role, the two intents of this identifier, one as node, and one as role, will never be confused in our theory, as we will never compare roles with nodes.

2.1. Informal meaning of edges

OPM is not some kind of programming language and OPM graphs are neither some kind of programs. Indeed, provenance information is not restricted to computer programs. Moreover, if one must insist on a programming analogy, then OPM graphs can better be thought of as representing *execution traces* of programs, without any further information about the program itself.

Indeed, the reference specification makes clear that edges in an OPM graph indicate causal dependencies. Specifically, the specification provides the following informal meanings:

edge type	edge	meaning
used	$P \xrightarrow{!} A$	P could not have completed without A
generated-by	$A \xrightarrow{!} P$	A could not have existed without P
informed-by	$P_2 \rightarrow P_1$	P_2 could not have completed without P_1
derived-from	$A_2 \rightarrow A_1$	A_2 could not have existed without A_1

Thus, for example, in Figure 1, we see that 'Third Party Process' must have ended after 'Take Order' was started; that 'toy' must have been produced after 'order' was given; that 'e-book' must have been produced after 'Deliver' started; and that 'Deliver' must have ended after the 'invoice info' was produced.

2.2. Inference rules

Interpreting edges in an OPM graph as causal dependencies, the OPM reference specification also includes four inference rules that infer new dependencies from the given dependencies in an OPM graph. Concretely, the rules infer new, so-called "multi-step" edges, denoted as $X \xrightarrow{*} Y$, of each of the four possible types. The rules are recursive and essentially based on the transitive closure of derived-from edges. Let A, B, C be artifacts and let P and Q be processes:

Basis. For every existing edge $X \xrightarrow{!} Y$ or $X \rightarrow Y$, we can infer the trivial multi-step edge $X \xrightarrow{*} Y$.
Derived-from. If we can infer $A \xrightarrow{*} B$ and $B \xrightarrow{*} C$, then we can also infer $A \xrightarrow{*} C$.
Generated-by. If we can infer $A \xrightarrow{*} B$ and $B \xrightarrow{*} P$, then we can also infer $A \xrightarrow{*} P$.
Used. If we can infer $P \xrightarrow{*} A$ and $A \xrightarrow{*} B$, then we can also infer $P \xrightarrow{*} B$.
Informed-by. If we can infer $P \xrightarrow{*} A$ and $A \xrightarrow{*} Q$, then we can also infer $P \xrightarrow{*} Q$.

For example, in Figure 1, we can infer multi-step edges from 'e-book' to 'order', from 'Deliver' to 'billing address', and from 'Deliver' to 'Take Order', among others. Note that we cannot infer a multi-step edge from 'delivery request' to 'billing address'.

Remark 2.4. Note that there are no further rules such as 'from $A \rightarrow P \rightarrow B$ infer $A \xrightarrow{*} B$', or 'from $P \rightarrow Q \rightarrow R$ infer $P \xrightarrow{*} R$'. We will see in Remark 4.3 that both of these inferences are unsound.

2.3. OPM: a critique from a formal viewpoint

Although the edges in an OPM graph have a clear informal meaning, a formal semantics is lacking. By developing such a formal semantics, we are able to address the following open issues, pertaining to edges, but also to other concepts underpinning OPM.

(1) A formal semantics can define what is correct reasoning in an OPM graph (cf. Remark 2.4), and would allow us to assess whether the inference rules are sound, or complete.
(2) Furthermore, there seems to be no good reason why multi-step edges can only be inferred, but cannot be asserted in the graph to begin with. For example, at a coarser level of granularity, it may be that only multi-step information is available without that the steps in between can be detailed. This issue was brought up by the "OPM community"[6], but it has not been investigated so far.
In this respect, the distinction between imprecise edges and multi-step edges is unclear. For example, in Figure 1, the edges 'toy' to 'order' and 'Third Party Process' to 'Take Order' have a multi-step flavor already. Conversely, one may wonder why there are no precise derived-from (or informed-by) edges, and why there are no imprecise used or generated-by edges. In general, the distinction between precise edges (edges carrying roles) and imprecise edges has been questioned during the development of OPM.[7]
(3) Time is regarded as a fundamental concept of OPM (and hence was voted to be kept in the OPM reference specification[8]), but the manner in which time contributes to the essence of OPM has not been established formally.
(4) OPM graphs containing cycles of derived-from edges are not legal according to the OPM reference specification. OPM also defines some operations over graphs, but some, such as union, are known to be capable of forming cycles, and hence leading to graphs that are not legal. Whether the acyclic nature of graphs should be mandated by default, or whether it can be inferred, and the conditions under which it holds is an open problem.
(5) An account is a construct by which multiple descriptions of execution can co-exist in a same OPM graph. The reference specification indicates that accounts may be related according to some relations, and suggests a notion of refinement. This notion again has not been formalized.

In the next two sections, we will address these gaps by proposing an extended model that allows imprecise edges asserted in a graph for all four types of edges; proposing a formal semantics that allows a rigorous notion of inference; and investigating sound and complete inference in OPM graphs. Our extended model allows precise derived-from edges, leading to a new notion of "use–generate–derive" triangles in OPM graphs that show a tight relation between a derivation and the process responsible for the derivation.

We believe the semantic distinction between imprecise and precise edges is important. Imprecise specifications relieve the provenance asserter of the extra burden of having to give precise derivations when these are not available or are not important for the application at hand; at the same time, when precise information is available and important, our theory allows the understanding of the extra knowledge that is provided.

3. OPM GRAPHS AND THEIR TEMPORAL SEMANTICS

We begin with an improved definition of OPM graph where both precise and imprecise edges of all four types are allowed, with the exception of precise informed-by edges.[9]

[6]See discussions in http://twiki.ipaw.info/bin/view/OPM/WorkInProgressV1pt1
[7]http://twiki.ipaw.info/bin/view/Challenge/FirstOPMWorkshopMinutes
[8]http://twiki.ipaw.info/bin/view/OPM/ChangeProposalMoveTimeToProfile
[9]We could have allowed precise informed-by edges, but in our current approach they would have exactly the same semantics as imprecise ones, which is why we omit them. It is an interesting open problem whether

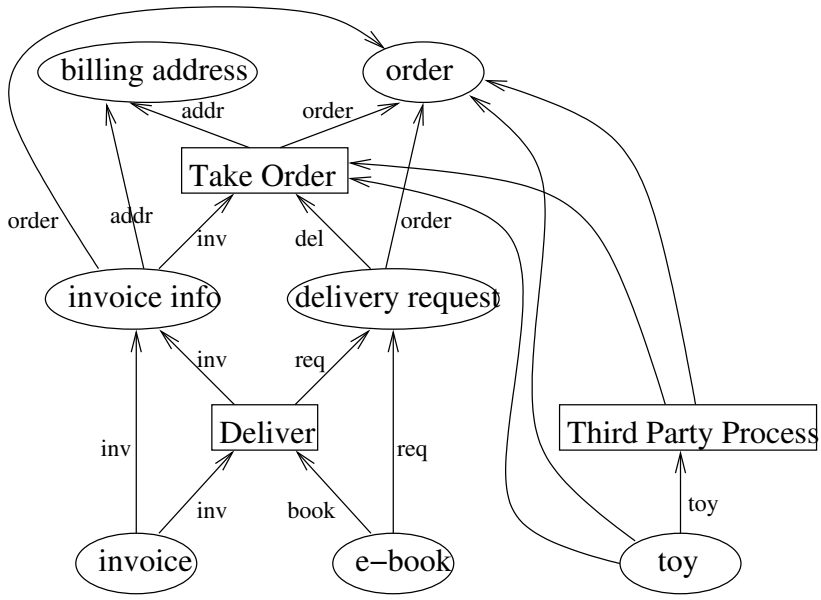

Fig. 2. Example of an OPM graph according to the improved definition.

Definition 3.1 (*OPM graph, improved definition*). An OPM graph is a structure

$$(Art, Proc, Roles, GeneratedBy!, Used!, DerivedFrom!, \\ GeneratedBy, Used, DerivedFrom, InformedBy)$$

where

— Art and $Proc$ are two disjoint finite sets of elements called *artifacts* and *processes*, respectively;
— $Roles$ is a finite set of elements called *roles*;
— $GeneratedBy! \subseteq Art \times Roles \times Proc$;
— $Used! \subseteq Proc \times Roles \times Art$;
— $DerivedFrom! \subseteq Art \times Roles \times Art$;
— $GeneratedBy \subseteq Art \times Proc$;
— $Used \subseteq Proc \times Art$;
— $DerivedFrom \subseteq Art \times Art$;
— $InformedBy \subseteq Proc \times Proc$.

In extension of our original terminology, the elements of $GeneratedBy! \cup Used! \cup DerivedFrom!$ are called *precise edges*, and the elements of $GeneratedBy \cup Used \cup DerivedFrom \cup InformedBy$ are called *imprecise edges*; all together they are called edges.

For future use, we introduce the following notation. When the distinction between precise and imprecise derived-from edges is of no consequence, we use the following set to refer to all derived-from edges of an OPM graph:

$$DerivedEdges = DerivedFrom \cup \{(A, B) \mid (A, r, B) \in DerivedFrom!\} .$$

there is a natural approach by which precise informed-by edges can be given a more specific semantics than imprecise ones.

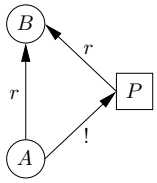

Fig. 3. A use–generate-derive triangle (A, B, P, r).

Example 3.2. The graph shown in Figure 2 extends the graph of Figures 1 with some new features allowed by the improved definition. All derived-from edges involved in the making of the invoice and e-book have now become precise. Moreover we have added an imprecise used-edge from 'Third Party Process' to 'order' and an imprecise generated-by edge from 'toy' to 'Take Order'.

3.1. Legality

The OPM reference specification defines a notion of legal graph as a directed graph without cycles in the derived-from edges, in which each artifact is generated by at most one process. It turns out that cycles in the derived-from edges do not strictly need to be forbidden; we will discuss this issue in more detail in Section 5.2. On the other hand, we can refine the notion of legality to take advantage of the new possibility of precise derived-from edges. This leads to the following.

Definition 3.3 (*Legal OPM graph*). An OPM graph is called *legal* if

— for each artifact A there is at most one process P with a precise generated-by edge $A \xrightarrow{!} P$; and
— for each precise derived-from edge $A \xrightarrow{r} B$ there is a process P with precise edges $A \xrightarrow{!} P$ and $P \xrightarrow{r} B$, for the same role r.

A configuration (A, B, P, r) with edges $A \xrightarrow{r} B$, $A \xrightarrow{!} P$, and $P \xrightarrow{r} B$, is called a *use–generate–derive triangle*, or simply *triangle* for short (see Figure 3). To denote that a use–generate–derive triangle (A, B, P, r) occurs in some given OPM graph G, we use the notation $G \triangle (A, B, P, r)$.

A use–generate–derive triangle is the graphical syntax we introduce to state in OPM that B was not merely used by P, but was specifically used as part of the creation of A. Thus, the common role r labeling both the usage of B by P and the derivation of A from B is used to express that this usage of B has affected the output A. Note that P might also use B in additional roles different from r, which would correspond to usages not relevant to the generation of A. The generated-by edge from A to P must be precise, but is not required to be labeled r, thus allowing more freedom for producers of OPM graphs. The generated-by edge must still be precise, failing which we would not be certain that it was P itself that generated A. This indeed will be the distinction in semantics between precise and imprecise generated-by edges introduced in Definition 3.6.

Example 3.4. Figure 2 contains a number use–generate–derive triangles, notably:

— Two triangles involving the process 'Deliver':
  — (invoice, invoice info, Deliver, inv);
  — (e-book, delivery request, Deliver, req).
— three more involving the process 'Take Order':
  — (invoice info, billing address, Take Order, addr);
  — (invoice info, order, Take Order, order);
  — (delivery request, order, Take Order, order). □

In this paper, unless otherwise explicitly stated, we only consider legal OPM graphs. Whenever we refer to a single OPM graph G, we use the names defined in this section to refer to the different constituents of the OPM graph. If we handle more than one OPM graph, for instance graphs G and H, we use superscripts G and H to distinguish their respective constituents. We extend this convention to other concepts related to OPM graphs.

3.2. Temporal models for OPM graphs

The informal meaning of OPM edges, given in Section 2, is temporal in nature. In order to formalize these statements, we need to refer to such temporal events as the production of an artifact, and the start and completion of a process. In order to better exploit precise use-edges and use–generate–derive triangles, we also refer to the *use* of an artifact by a process as a temporal event. This leads us to introduce the formal notion of temporal model which we develop in this section.[10]

We start with the notion of a *temporal interpretation* of a legal OPM graph: an assignment of time-points to processes, artifacts, and precise used-edges, specifically:[11]

— for each artifact A, its creation time, denoted by $\mathrm{create}(A)$;
— for each process P, its beginning and ending times, denoted by $\mathrm{begin}(P)$ and $\mathrm{end}(P)$ respectively;
— for each precise used-edge $P \xrightarrow{r} A$, the time when P used A in role r, denoted by $\mathrm{use}(P, r, A)$.

To make this formal, we fix some OPM graph G for the remainder of this section. We define the set of *temporal variables* of G, denoted by $\mathit{Vars}(G)$, as follows:

$$\mathit{Vars}(G) = \{\mathrm{create}(A) \mid A \in \mathit{Art}\} \cup \{\mathrm{begin}(P), \mathrm{end}(P) \mid P \in \mathit{Proc}\} \cup \{\mathrm{use}(P, r, A) \mid (P, r, A) \in \mathit{Used}!\}\,.$$

We then define:

Definition 3.5. A *temporal interpretation* of G is a triple $(T, \leq, \tau)$, where

— T is a set, we call its elements *time-points*;
— $\leq$ is a partial order on T;
— τ is a mapping from $\mathit{Vars}(G)$ to T.

When no confusion can arise, we omit T and $\leq$ from the notation and simply denote a temporal interpretation by τ.

Not every temporal interpretation makes sense as a temporal model of G. Indeed, to reflect the dependencies specified in G, the interpretation should satisfy various constraints reflecting these dependencies.

In order to define these constraints formally, we define an *inequality* over G as a syntactical expression of the form $u \preceq v$, with $u, v \in \mathit{Vars}(G)$. Thus, inequalities are simple formulas. By a *trivial* inequality we mean an inequality of the form $u \preceq u$. We are now ready to define the set of constraints expressed by a legal OPM graph.

[10] The reference specification also allows OPM graphs to be explicitly decorated with time points, but this is optional, so in this paper we focus on the semantics of "blank" graphs: graphs initially without temporal decorations. Indeed, the temporal models developed in this section correspond to all consistent ways in which such a blank OPM graph *can* be decorated.

[11] One may wonder why precise generated-by edges do not get a time-point. But, as a matter of fact, they do. For each precise generated-by edge $A \xrightarrow{r} P$, we indeed have a time-point $\mathrm{create}(A)$. Since the OPM graph is legal, there can be at most one precise edge emanating from A, so we do not need to specify r and P.

Table I. The inequalities making up the temporal theory of an OPM graph.

- AX 1: for each process P, the inequality $\text{begin}(P) \preceq \text{end}(P)$;
- AX 2: for each precise generated-by edge $A \xrightarrow{!} P$ in G, the inequalities $\text{begin}(P) \preceq \text{create}(A) \preceq \text{end}(P)$;
- AX 3: for each precise used-edge $P \xrightarrow{r} A$ in G, the three inequalities $\text{begin}(P) \preceq \text{use}(P, r, A)$, $\text{use}(P, r, A) \preceq \text{end}(P)$, and $\text{create}(A) \preceq \text{use}(P, r, A)$;
- AX 4: for each imprecise derived-from edge $A \to B$ in G, the inequality $\text{create}(B) \preceq \text{create}(A)$;
- AX 5: for each imprecise generated-by edge $A \to P$ in G, the inequality $\text{begin}(P) \preceq \text{create}(A)$;
- AX 6: for each imprecise used-edge $P \to A$ in G, the inequality $\text{create}(A) \preceq \text{end}(P)$;
- AX 7: for each informed-by edge $P \to Q$ in G, the inequality $\text{begin}(Q) \preceq \text{end}(P)$;
- AX 8: for each $G \triangle (A, B, P, r)$, the inequality $\text{use}(P, r, B) \preceq \text{create}(A)$.

Table II. Two temporal models for the graph shown in Figure 3.

τ_1	variable	value
	create(B)	1
	begin(P)	2
	use(P, r, B)	3
	create(A)	4
	end(P)	5

τ_2	variable	value
	create(B)	1
	begin(P)	1
	use(P, r, B)	1
	create(A)	1
	end(P)	1

Definition 3.6. The *temporal theory* of G, denoted by $\text{Th}(G)$, is the set consisting of all the inequalities stated in the *axioms* listed in Table I.[12]

Axioms 1–3 are clear; and Axioms 4–7 are a verbatim formalization of the informal meaning of edges from the OPM reference specification, recalled in Section 2. The eighth and final axiom, the "triangle axiom", corresponds to the intended usage of OPM by which a precise derived-from edge in a use–generate–derive triangle (A, B, P, r) in G is not redundant, but expresses exactly that P needed to read B in role r before it could generate A.

Example 3.7. In Figure 2, the precise generated-by edge from 'e-book' to 'Deliver' yields the constraint that the e-book access code was made available before the 'Deliver' process completed. In contrast, the imprecise generated-by edge from 'toy' to 'Take Order' only yields that the toy was delivered by the third party after the 'Take Order' process started. The use–generate–derive triangle (e-book, delivery request, Deliver, req) also yields that the delivery request was received by the Deliver process before the e-book access code was communicated. □

We finally define the temporal models of G as follows. Naturally, a temporal interpretation τ is said to *satisfy* an inequality $u \preceq v$ if $\tau(u) \leq \tau(v)$.

Definition 3.8. A temporal interpretation τ of G is a *temporal model* of G, denoted by $\tau \models \text{Th}(G)$, if it satisfies all inequalities from $\text{Th}(G)$.

Example 3.9. Consider the small OPM graph G shown in Figure 3. Let us use natural numbers with their natural ordering as time-points. Then the two interpretations τ_1 and τ_2, presented in Table II, are temporal models of G. Temporal model τ_2, which maps all temporal variables to the same time-point, might be generated by a very coarse clock.

Many temporal interpretations of G, however, are not temporal models of G. If, for example, we would modify τ_1 to τ_1' by setting $\tau_1'(\text{end}(P)) = 0$, then Axiom 1 would be violated. Likewise, if we would modify τ_2 to τ_2' by setting $\tau_2'(\text{use}(P, r, B)) = 0$, then Axiom 3 would be violated. Also, if we would modify τ_1 to τ_1'' by setting $\tau_1''(\text{create}(A)) = 0$, then we would violate Axioms 2 and 8. □

[12] In mathematical logic, a theory is sometimes (but not always) defined as a set of formulas closed under logical implication. Here it is just a set of formulas, not necessarily closed in this way.

Example 3.10. For another example, consider an OPM graph with two artifacts A and B and one process P, with edges $A \xrightarrow{!} P \xrightarrow{r} B$. Then both τ_1 and τ_2, defined as follows, are models of the graph:

τ_1	
create(B)	1
begin(P)	2
use(P, r, B)	3
create(A)	4
end(P)	5

τ_2	
begin(P)	1
create(A)	2
create(B)	3
use(P, r, B)	4
end(P)	5

In model τ_1, P uses B before creating A, but in model τ_2, B does not yet exist when A is created. Note that B does not have a generated-by edge, so the graph does not specify which process generated B. □

Example 3.11. Consider a graph with three processes P, Q and R and edges $P \to Q \to R$. Then both τ_1 and τ_2, defined as follows, are models of the graph:

τ_1	
begin(R)	1
end(R)	2
begin(Q)	3
end(Q)	4
begin(P)	5
end(P)	6

τ_2	
begin(P)	1
begin(Q)	2
end(P)	3
begin(R)	4
end(Q)	5
end(R)	6

□

Example 3.12. For a final example, consider an OPM graph with two artifacts A and B and nothing else (no edges either). Then any possible temporal interpretation qualifies as a model. In particular, in some models τ we have $\tau(\text{create}(A)) < \tau(\text{create}(B))$; in other models we have $\tau(\text{create}(B)) < \tau(\text{create}(A))$; and still in others we have $\tau(\text{create}(A)) = \tau(\text{create}(B))$. This is because the OPM graph does not impose any constraints due to the absence of any edge between A and B. □

4. INFERENCE IN OPM GRAPHS

The axioms of Definition 3.6 allow us to obtain a number of inequalities over an OPM graph's variables. These inequalities logically imply further inequalities. For a trivial example, in an OPM graph with derived-from edges $A \to B \to C$, Axiom 4 gives the inequalities $\text{create}(C) \preceq \text{create}(B)$ and $\text{create}(B) \preceq \text{create}(A)$, which logically imply the further inequality $\text{create}(C) \preceq \text{create}(A)$.

Formally, we define:

Definition 4.1. Let G be a legal OPM graph and let $u, v \in \mathit{Vars}(G)$. The inequality $u \preceq v$ is a *logical consequence* of G, denoted by $\text{Th}(G) \models u \preceq v$, if $u \preceq v$ is satisfied in every temporal model of G.

A general example of logical consequence is provided by the following lemma and proof.

LEMMA 4.2. *Let G be a legal OPM graph with artifacts A and B and a precise edge $A \xrightarrow{r} B$ for some role r. Then* $\text{Th}(G) \models \text{create}(B) \preceq \text{create}(A)$.

Before proving this lemma we note that Axiom 4 is almost exactly the same, except that it is stated for an imprecise derived-from edge. Thus, the present lemma shows that the same constraint holds for precise derived-from edges. This constraint did not need to be explicitly given as an axiom because it already logically follows from the given axioms.

PROOF. Since G is legal, there exists a process P in G with edges $P \xrightarrow{r} B$ and $A \xrightarrow{!} P$. Let τ be a temporal model of G. By Axiom 3 we have $\tau(\text{create}(B)) \leq \tau(\text{use}(P, r, B))$. By Axiom 2 we have $\tau(\text{use}(P, r, B)) \leq \tau(\text{create}(A))$. We conclude $\tau(\text{create}(B)) \leq \tau(\text{create}(A))$ as desired. □

Remark 4.3. Model τ_2 from Example 3.10 shows that $\text{create}(B) \preceq \text{create}(A)$ is *not* a logical consequence of a graph with edges $A \to P \to B$. Since $\text{create}(B) \preceq \text{create}(A)$ is the semantics of an (imprecise) derived-from edge (Axiom 4), this explains the unsoundness of the first inference considered in Remark 2.4. Similarly, model τ_2 from Example 3.11 shows that the $\text{begin}(R) \preceq \text{end}(P)$ is *not* a logical consequence of a graph with edges $P \to Q \to R$, thus explaining the unsoundness of the second inference considered in Remark 2.4. □

A fundamental problem is to know exactly which inequalities logically follow from a given OPM graph. This problem has many applications. For example, if the provenance information for the US National Climate Assessment is recorded carefully [Tilmes et al. 2013], we may want to make sanity checks such as inferring that some source data must have been generated before various conclusions were drawn from them.

It is well known that an inequality $u \preceq w$ can be inferred from $\text{Th}(G)$ by using repeated applications of the rule of transitivity: "from $u \preceq v$ and $v \preceq w$ we infer $u \preceq w$".[13] Such an approach is unsatisfactory, however, as it is hard to relate the newly inferred inequalities to nodes and edges in the graph.

The inference given by Lemma 4.2 provides a case in point. When the user asks why $\text{create}(B) \preceq \text{create}(A)$ must hold, pure reasoning by inequalities can only present the following proof:

$$\begin{aligned} \text{create}(B) &\preceq \text{use}(P, r, B) && \text{by axiom 3} \\ &\preceq \text{create}(A) && \text{by axiom 2} \end{aligned}$$

In contrast, a much more direct justification for $\text{create}(B) \preceq \text{create}(A)$ is the presence of the edge $A \xrightarrow{r} B$ in the graph.

Indeed, we show in Section 4.2 that it is always possible to provide such direct justifications, by performing temporal inference in a purely graphical manner. We prove in Theorem 4.7 that every possible logical consequence can be directly inferred from the OPM graph by looking for a fixed set of patterns in the graph.

4.1. Edge-inference rules

The cornerstone of our graph-based inference of inequalities is provided by the four original OPM inference rules already recalled in Section 2. We revisit them here in full detail, at the same time extending them so as to better exploit the possible presence of precise edges. Inferred edges will prove to play an important role in graphical patterns that we introduce to infer inequalities. Moreover, we establish that inference of edges is the only action we need to perform to infer inequalities that do not involve use-variables. (For inequalities involving use-variables, patterns more complicated than a single edge have to be matched in the graph.) We thus provide a justification for the edge inferences introduced in the OPM reference specification.

We first argue for the inference of edges at the intuitive level, based on following chains of derived-from edges. Then we define edge inference formally in Definition 4.5.

Suppose there is a chain of derived-from edges in G (which can be either precise or imprecise) that starts in an artifact A and ends in an artifact C. We denote this by $A \dashrightarrow C$.

[13] For a set of inequalities Σ and an inequality φ, φ is a logical consequence of Σ if and only if φ can be inferred from Σ by using transitivity. Ullman [Ullman 1989] presents a self-contained proof for a slightly more general case.

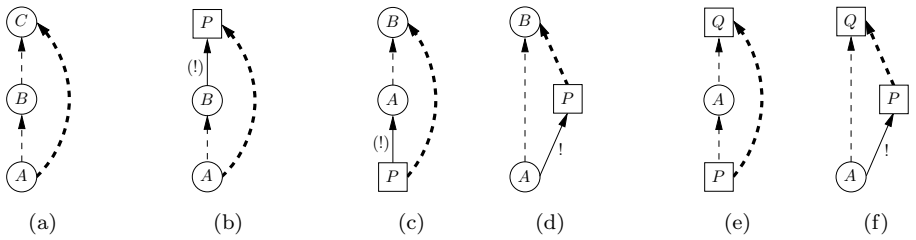

Fig. 4. Inference of (a) derived-from, (b) generated-by, (c)–(d) used and (e)–(f) informed-by edges. The bold edges are newly inferred. The edges labeled by "(!)" may be either precise or imprecise.

Formally, relation $\dashrightarrow$ between two artifacts is nothing else than the transitive closure of *DerivedEdges*. Since A has been indirectly derived from C, we can think of $A \dashrightarrow C$ as an inferred edge, as illustrated in Figure 4(a).

Next we show how to infer generated-by edges. Suppose we have artifacts A and B with $A \dashrightarrow B$ and, in addition, a generated-by edge from B to a process P in G, either a precise edge $B \xrightarrow{!} P$ or an imprecise edge $B \rightarrow P$. Then A has been indirectly generated by P and we can infer an edge $A \dashrightarrow P$, as illustrated in Figure 4(b).

We can infer used-edges as well. Suppose we have artifacts A and B with $A \dashrightarrow B$. In addition, there is a used-edge from a process P to A in G, either precise $P \xrightarrow{!} A$ or imprecise $P \rightarrow A$. Then P has indirectly used B and we can infer an edge $P \dashrightarrow B$, as illustrated in Figure 4(c). Moreover, we can also infer a used-edge in the following situation. Suppose we again have $A \dashrightarrow B$, but now in combination with a precise edge $A \xrightarrow{!} P$ in G. Since A was precisely generated by P, but A has also been indirectly derived from B, we can conclude that P has indirectly used B. Again, we can infer $P \dashrightarrow B$, which we show in Figure 4(d).

Finally, to infer informed-by edges, we can reason as follows. Suppose, for some processes P and Q and an artifact A, we have edges $P \dashrightarrow A$ and $A \dashrightarrow Q$. Then, A represents information that flowed from Q to P and we can infer an edge $P \dashrightarrow Q$, as illustrated in Figure 4(e). Moreover, an informed-by edge can also be inferred in the following case. Suppose we again have $A \dashrightarrow Q$, but now in combination with a precise edge $A \xrightarrow{!} P$ in G. Since A was directly generated by P, but A was also indirectly generated by Q, we can conclude that P was somehow influenced by Q. Again, we can infer $P \dashrightarrow Q$, which we show in Figure 4(f).

There are also trivial inferences for all types of edges, to the effect that an edge that is already present in the graph can always be inferred.

Example 4.4. In Figure 2, we can now see that the informed-by edge from 'Third Party Process' to 'Take Order' is redundant, in the sense that it can already be inferred using the precise edge from 'toy' to 'Third Party Process' and the imprecise edge from 'toy' to 'Take Order' (Figure 4(f).) Likewise, the imprecise used edge from 'Third Party Process' to 'order' is redundant in the sense that it can be inferred now using the edge from 'toy' to 'order' (Figure 4(d).) □

The above discussion is formalized in the following definition. We present the rules in a standard notation used in formal logic, where for each rule the premises are stated above a bar, and the conclusion below it.

Definition 4.5 (*Edge-inference rules*). Let G be a legal OPM graph and let X and Y be two nodes in G. In the following, we define when $X \dashrightarrow Y$ can be *inferred* from G, denoted by $G \vdash X \dashrightarrow Y$. Specifically, let A, B and C be artifacts in G, and let P and Q be processes in G.

$$\frac{A \to B \text{ in } G \text{ or } A \overset{!}{\to} B \text{ in } G}{G \vdash A \dashrightarrow B} \text{ TRIVIAL DERIVED-FROM}$$

$$\frac{A \to P \text{ in } G \text{ or } A \overset{!}{\to} P \text{ in } G}{G \vdash A \dashrightarrow P} \text{ TRIVIAL GENERATED-BY}$$

$$\frac{P \to A \text{ in } G \text{ or } P \overset{!}{\to} A \text{ in } G}{G \vdash P \dashrightarrow A} \text{ TRIVIAL USED} \qquad \frac{P \to Q \text{ in } G}{G \vdash P \dashrightarrow Q} \text{ TRIVIAL INFORMED-BY}$$

Fig. 5. Trivial edge-inference rules.

$$\frac{G \vdash A \dashrightarrow B \qquad G \vdash B \dashrightarrow C}{G \vdash A \dashrightarrow C} \text{ DERIVED-FROM}$$

$$\frac{G \vdash A \dashrightarrow B \qquad B \to P \text{ in } G \text{ or } B \overset{!}{\to} P \text{ in } G}{G \vdash A \dashrightarrow P} \text{ GENERATED-BY}$$

$$\frac{G \vdash A \dashrightarrow B \qquad P \to A \text{ in } G \text{ or } P \overset{!}{\to} A \text{ in } G \text{ or } A \overset{!}{\to} P \text{ in } G}{G \vdash P \dashrightarrow B} \text{ USED}$$

$$\frac{G \vdash A \dashrightarrow Q \qquad G \vdash P \dashrightarrow A \text{ or } A \overset{!}{\to} P \text{ in } G}{G \vdash P \dashrightarrow Q} \text{ INFORMED-BY}$$

Fig. 6. Edge-inference rules.

We begin by stating four trivial inference rules which mean that if an edge already belongs to G, then that edge can be inferred from G. These rules are presented in Figure 5. Next we define four further inference rules, in cases where at least one of the present edges was previously inferred. These rules are presented in Figure 6.

Note that, as a direct consequence of the above definition, we have the following properties:

— $G \vdash A \dashrightarrow B$ iff (A, B) belongs to the transitive closure of *DerivedEdges*;
— if $G \vdash A \dashrightarrow B$ and $G \vdash B \dashrightarrow P$ then $G \vdash A \dashrightarrow P$;
— if $G \vdash P \dashrightarrow A$ and $G \vdash A \dashrightarrow B$ then $G \vdash P \dashrightarrow B$.

Edge-inference rules introduced in this section allow us to derive new edges from a graph G, noted as $G \vdash X \dashrightarrow Y$, with X and Y two nodes of G. Inferred edges do not belong to the sets of edges identified in Definition 3.1, implying that these edges $X \dashrightarrow Y$ are inferred "outside" G. Thus, the temporal theory of Definition 3.6 does not associate a temporal meaning to these edges, directly. In the next section, we observe that inferred edges have a similar temporal semantics as imprecise edges.

4.2. Characterization of temporal inference

Let us reconsider the axioms of Definition 3.6 that define the temporal semantics of an OPM graph. We see that each axiom is a rule that relates a pattern in the graph to one or more inequalities. For example, Axiom 2 relates the pattern consisting simply of a single

edge $A \xrightarrow{!} P$, to the inequalities $\text{begin}(P) \preceq \text{create}(A)$ and $\text{create}(A) \preceq \text{end}(P)$. Axiom 1 even relates the pattern consisting simply of a process node P to the inequality $\text{begin}(P) \preceq \text{end}(P)$. Axiom 8 has a more complicated pattern in the form of a use–generate–derive triangle.

In a similar way, we now introduce ten more such rules. Rules 1–9A–9B are shown in Figure 7. (The figure also includes some axioms, but we explain this after the statement of Theorem 4.7.) An important difference with the axioms, however, is that every dashed edge in a pattern now stands not just for an edge that is present in the graph, but for an edge that can be inferred by the edge-inference rules.

Example 4.6. In Figure 2, let u_1 be the timepoint where 'Take Order' uses the 'billing address', and let u_2 be the timepoint where 'Deliver' uses the 'invoice info'. Then rule 9A derives that $u_1 \preceq u_2$. This can be seen by matching node Q in the rule to the node 'Deliver' in the graph; A to 'invoice info'; B to 'billing address'; and P to 'Take Order'. □

The following theorem, proven in the Appendix, states that these rules are *sound* and *complete* in the following sense. The rules are sound in that they represent valid inferences: the inequalities they infer are indeed logical consequences of the axioms in the sense of Definition 4.1. Moreover, the rules are complete in that any inequality that is a logical consequence of the axioms, and that is not already part of the axioms, can be inferred by one of the ten rules.

THEOREM 4.7. *Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G. Then* $\text{Th}(G) \models \varphi$ *if and only if either (0) φ already belongs to* $\text{Th}(G)$*, or φ matches one of the following inequalities:*

— *Cases* not *involving use-variables:*
- (*1*) $\text{create}(B) \preceq \text{create}(A)$ *with* $G \vdash A \dashrightarrow B$*;*
- (*2*) $\text{begin}(P) \preceq \text{create}(A)$ *with* $G \vdash A \dashrightarrow P$*;*
- (*3*) $\text{create}(A) \preceq \text{end}(P)$ *with* $G \vdash P \dashrightarrow A$*;*
- (*4*) $\text{begin}(Q) \preceq \text{end}(P)$ *with* $G \vdash P \dashrightarrow Q$*;*

— *Cases involving use-variables:*
- (*5*) $\text{create}(B) \preceq \text{use}(P, r, A)$ *with* $P \xrightarrow{r} A$ *in* G *and* $G \vdash A \dashrightarrow B$*;*
- (*6*) $\text{begin}(Q) \preceq \text{use}(P, r, A)$ *with* $P \xrightarrow{r} A$ *in* G *and* $G \vdash A \dashrightarrow Q$*;*
- (*7*) $\text{use}(P, r, C) \preceq \text{create}(A)$ *with* $G \triangle (B, C, P, r)$ *for some* B*, and* $G \vdash A \dashrightarrow B$*;*
- (*8*) $\text{use}(P, r, B) \preceq \text{end}(Q)$ *with* $G \triangle (A, B, P, r)$ *for some* A*, and* $G \vdash Q \dashrightarrow A$*;*
- (*9*) $\text{use}(P, r, B) \preceq \text{use}(Q, s, A)$ *with* $G \triangle (C, B, P, r)$ *for some* C*, with* $Q \xrightarrow{s} A$ *in* G*, and either (a)* $A = C$ *or (b)* $G \vdash A \dashrightarrow C$*.*

Note that in the above, A, B and C, or P and Q, need not be distinct.

Since Rules 1–4 subsume Axioms 4–7, Figure 7, which includes the remaining axioms, provides a complete picture of the possible logical consequences of an OPM graph in the sense of Definition 4.1. Definition 4.1 is purely semantic and does not give any concrete algorithm for checking logical consequence. Figure 7 now gives us direct shortcuts from patterns in an OPM graph to its logical consequences. Indeed, to check that an inequality $u \preceq v$ is logical consequence of a graph, it suffices to select the corresponding pattern in Figure 7, and verify that it is satisfied by the graph (extended with the proper inferred edges). Vice versa, if an inequality $u \preceq v$ is logical consequence of $\text{Th}(G)$, then the corresponding pattern is known to exist in G.

We anticipate that developers can leverage Theorem 4.7 to design reasoners for OPM. So far, reasoners have typically relied on Semantic Web technologies, such as OWL and SRWL, to compute transitive closures of OPM properties [Moreau et al. 2010; McGrath and Futrelle 2008]. What this theorem shows is that there are logical consequences involving

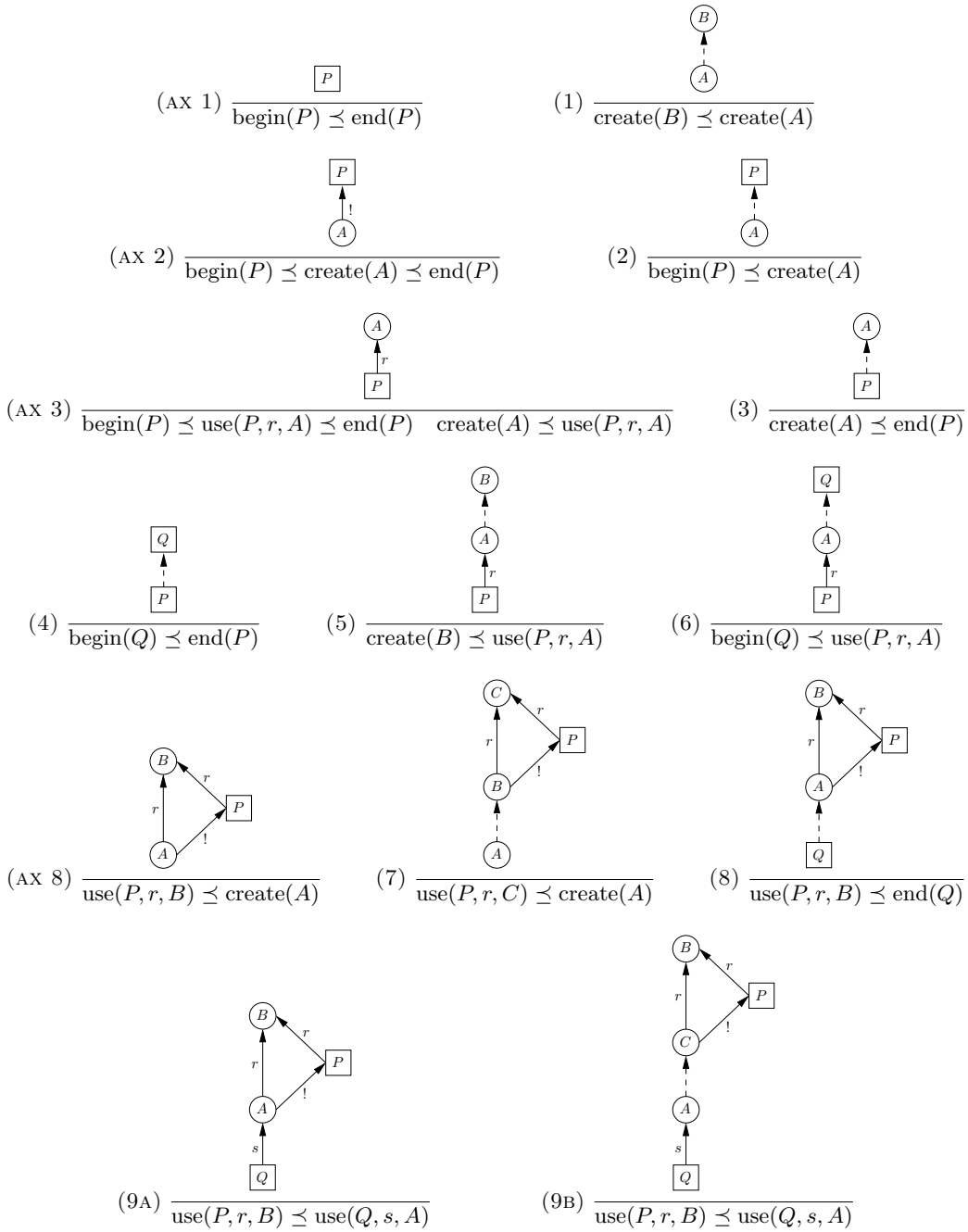

Fig. 7. Characterization of temporal inference. The numbers (1) to (9a) and (9b) correspond to the rules numbered (1) to (9a) and (9b) stated in Theorem 4.7. Since the Theorem also invokes, as rule (0), the temporal theory of G, we also show the axioms of $\mathrm{Th}(G)$ from Definition 3.6, so that this figure gives a complete picture of logical consequence. Note, however, that Axioms 4 to 7 from Definition 3.6 need not be shown because they are already subsumed by rules (1) to (4). Hence, only Axioms 1, 2, 3 and 8 are shown. For inference of inequalities not involving use-variables, Axioms 1–2 and rules 1–4 already suffice; this will be shown in Section 4.3.

use-variables that cannot be directly represented by a single edge in OPM graphs; rather, they are represented by graph patterns involving three or more nodes, as shown in Figure 7.

4.3. About non-use inequalities

The Open Provenance Model reference specification defines edges incident to artifacts in terms of the creation of the artifact, with respect to the creation of another artifact, or the beginning and ending of a process. There is some value in considering a temporal theory that ignores 'use' time-points, since the theory becomes simpler (though it is unable to tell us anything about usage of artifacts). In this case, it is worth characterizing temporal inference in the context of this simpler theory.

First we state a remarkable corollary, after introducing the following definition.

Definition 4.8. If in an inequality φ of the form $u \preceq v$, neither u nor v is a use-variable, then we call φ a *non-use* inequality.

As a corollary to Theorem 4.7, we obtain the following completeness result for edge inference, as far as non-use inequalities are concerned. Note that the edge-inference rules are present as Rules 1–4 in Figure 7.

COROLLARY 4.9. *Let G be a legal OPM graph and let φ be a non-use inequality. Then $\mathrm{Th}(G) \models \varphi$ if and only if φ can be inferred using Axioms 1–2 and Rules 1–4 in Figure 7.*

PROOF. It is clear from Theorem 4.7 that if φ can be inferred using Axioms 1–2 and Rules 1–4, then $\mathrm{Th}(G) \models \varphi$. For the other direction, assume that $\mathrm{Th}(G) \models \varphi$ holds. Then we know by Theorem 4.7 that φ can be inferred by the axioms and rules presented in Figure 7. By examination of these axioms and rules, however, we notice that Axioms 1–2 and Rules 1–4 are the only ones that infer non-use inequalities. □

It is interesting to note, that when dealing with non-use inequalities, we do not need the full temporal theory of an OPM graph. We start with a small generalization of Definitions 3.8 and 4.1.

Definition 4.10. Let G be a legal OPM graph and let $u, v \in \mathit{Vars}(G)$. Let Σ be a subset of $\mathrm{Th}(G)$. Any temporal interpretation that satisfies all inequalities of Σ is called a *temporal model* of Σ. Furthermore, the inequality $u \preceq v$ is a *logical consequence* of Σ, denoted by $\Sigma \models u \preceq v$, if $u \preceq v$ is satisfied in every temporal model of Σ.

For a given OPM graph G, we can now select the non-use inequalities from its temporal theory.

Definition 4.11. For a legal OPM graph G, we define the *non-use temporal theory* of G, denoted by $\mathrm{Th}^{\text{non-use}}(G)$, as follows:

$$\begin{aligned}\mathrm{Th}^{\text{non-use}}(G) = \{\varphi \in \mathrm{Th}(G) \mid \varphi \text{ is a non-use inequality}\} \\ \cup \left\{\mathrm{create}(A) \preceq \mathrm{end}(P) \mid P \xrightarrow{!} A \text{ in } G\right\} \\ \cup \left\{\mathrm{create}(B) \preceq \mathrm{create}(A) \mid A \xrightarrow{!} B \text{ in } G\right\}.\end{aligned}$$

The intuition is that $\mathrm{Th}^{\text{non-use}}(G)$ does not contain Axioms 3 and 8, and enforces Axioms 4 and 6 for precise and imprecise edges alike.[14]

[14]Note that in the full theory $\mathrm{Th}(G)$, the non-use inequality $\mathrm{create}(A) \preceq \mathrm{end}(P)$ for $P \xrightarrow{!} A$ in G is implied by Axiom 3, but since we omit this axiom, we need to recover the inequality in Axiom 6. Likewise, the non-use inequality $\mathrm{create}(B) \preceq \mathrm{create}(A)$ for $A \xrightarrow{!} B$ in G is provided by Lemma 4.2. Since the proof of the lemma utilizes use-variables, the lemma doesn't hold anymore and we need to recover the inequality in Axiom 4.

We can now observe that use-variables do not influence the non-use inequalities that are logical consequences of $\mathrm{Th}(G)$.

PROPOSITION 4.12. *Let G be a legal OPM graph and let φ be a non-use inequality. Then $\mathrm{Th}(G) \models \varphi$ if and only if $\mathrm{Th}^{non\text{-}use}(G) \models \varphi$.*

PROOF. Since any temporal model τ of $\mathrm{Th}(G)$ is also a temporal model of $\mathrm{Th}^{\text{non-use}}(G)$, the if-direction is immediate. For the only-if direction, let τ be a temporal model of $\mathrm{Th}^{\text{non-use}}(G)$. We try to extend τ to τ' in such a way that τ' is a temporal model of $\mathrm{Th}(G)$. For every non-use variable u simply put $\tau'(u) = \tau(u)$. Now we have to find suitable values for:

— $\mathrm{use}(P, r, A)$ for each $P \xrightarrow{r} A$ in G, so that Axiom 3 is satisfied, and
— $\mathrm{use}(P, r, B)$ for each $G \triangle (A, B, P, r)$, so that Axiom 8 is satisfied.

It is easy to verify that Axiom 3 holds if $\tau'(\mathrm{use}(P, r, A))$ equals the maximum of $\tau(\mathrm{create}(A))$ and $\tau(\mathrm{begin}(P))$. Likewise, Axiom 8 holds if $\tau'(\mathrm{use}(P, r, B))$ equals the maximum of $\tau(\mathrm{create}(B))$ and $\tau(\mathrm{begin}(P))$. Thus τ' satisfies all eight axioms and is a temporal model of $\mathrm{Th}(G)$. We know thus that τ' satisfies φ. Since φ is a non-use inequality, and τ' and τ coincide on all variables used in non-use inequalities, τ also satisfies φ. □

The above proposition together with Corollary 4.9 yields the following:

COROLLARY 4.13. *Let G be a legal OPM graph and let φ be a non-use inequality. Then $\mathrm{Th}^{non\text{-}use}(G) \models \varphi$ if and only if φ can be inferred using Axioms 1–2 and Rules 1–4 in Figure 7.*

This section provides a remarkable result since it establishes the completeness of edge inferences (Rules 1–4 in Figure 7) for non-use inequalities. Furthermore, reasoning with use time-points does not allow us to derive any new inequality about non-use variables. We envisage this result to be leveraged by developers of reasoners for OPM, since it offers opportunities to optimize reasoners, by reducing the number of time-points to reason over, focusing on non-use variables in a first phase, and dealing efficiently with use-variables afterwards.

5. OPERATIONS ON OPM GRAPHS

The reason for capturing provenance is that it can be used to address a variety of use cases [Miles et al. 2007]. To this end, one needs to collect provenance information from potentially different sources across the Web, combine and process it in multiple ways. It is therefore useful to define operations on OPM graphs, which we anticipate can become part of "provenance toolkits".

When two OPM graphs are obtained from different sources, a reasoner may want to take their union, if it ascertains they relate to some common entities. Given two OPM graphs, an intersection operation helps identify their common elements. Different sources may use different identifiers for graph nodes; thus, to be able to compute meaningful union and intersection, it may be required to rename some nodes, before performing these operations. In this section, we formally investigate the effect of these operations on OPM graphs on the temporal theories of these graphs.

Initial observations on graph operations and legality

(1) A subgraph of a legal OPM graph may not be legal. For example, the graph presented in Figure 8(a) is legal, whereas its subgraph composed of nodes A, B, P, role r, and edges $A \xrightarrow{r} B$ and $A \xrightarrow{r'} P$ is not legal, since the use–generate–derive triangle (A, B, P, r) is not complete.

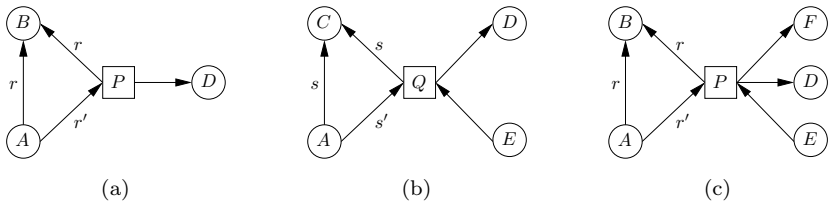

Fig. 8. Three legal OPM graphs.

(2) The union of two legal OPM graphs may not be legal. For instance, the graph presented in Figure 8(a) is legal, and so is the graph in Figure 8(b). The union of these two graphs, however, is not legal, since in the union A has two different precise generated-by edges: $A \xrightarrow{r'} P$ and $A \xrightarrow{s'} Q$.
(3) The intersection of two legal OPM graphs may not be legal. For instance, consider the graph G from Figure 10(a) and the graph H which equals G except that identifier P is replaced by a different identifier Q. Then $G \cap H$ consists only of the three artifacts A, B and D, and the single edge $A \xrightarrow{r} B$. Due to this edge (without accompanying use-generate-derive triangle) the intersection is not legal.

Union and intersection of theories. It is an interesting question for further research how to deal with non-legality of union, or intersection, of legal OPM graphs. Presumably a kind of "error-correcting" variants of union and intersection may be defined so that the result is always legal, and so that the temporal theory of the result has some desired properties. While we defer this question to further research, we can already answer some immediate questions about the relationship between legal union and intersection of legal OPM graphs and their temporal theories. Let G and H now be two legal OPM graphs.

PROPOSITION 5.1. *If $G \cup H$ is legal then* $\mathrm{Th}(G \cup H) = \mathrm{Th}(G) \cup \mathrm{Th}(H)$.

PROOF. Each inequality in $\mathrm{Th}(G)$ or $\mathrm{Th}(H)$ corresponds to a single node, a single edge or some use–generate–derive triangle present in G or H. Thus all inequalities present in $\mathrm{Th}(G) \cup \mathrm{Th}(H)$, also belong to $\mathrm{Th}(G \cup H)$. Moreover, the only additional inequalities in $\mathrm{Th}(G \cup H)$ would correspond to some use–generate-derive triangles that were newly formed by the union of G and H. Since both G and H are legal, this is impossible, because legal OPM graphs cannot contain parts of a use–generate-derive triangle. Therefore, $\mathrm{Th}(G \cup H)$ contains only inequalities that are already present in $\mathrm{Th}(G)$, or in $\mathrm{Th}(H)$, or in both. □

PROPOSITION 5.2. *If $G \cap H$ is legal then* $\mathrm{Th}(G \cap H) \subseteq \mathrm{Th}(G) \cap \mathrm{Th}(H)$.

PROOF. Any inequality from $\mathrm{Th}(G \cap H)$ corresponds to a single node, an edge, or a use–generate–derive triangle present in $G \cap H$, and thus in both G and H. Therefore, it also belongs to $\mathrm{Th}(G) \cap \mathrm{Th}(H)$. □

The converse inclusion does not hold. If G consists only of edge $P \to A$, and H consists only of edge $A \xrightarrow{!} P$, then $G \cap H$ consists of the two nodes A and P. So,

$$\mathrm{Th}(G) = \{\mathrm{create}(A) \preceq \mathrm{end}(P), \mathrm{begin}(P) \preceq \mathrm{end}(P)\} ,$$
$$\mathrm{Th}(H) = \{\mathrm{begin}(P) \preceq \mathrm{create}(A), \mathrm{create}(A) \preceq \mathrm{end}(P), \mathrm{begin}(P) \preceq \mathrm{end}(P)\} ,$$

and

$$\mathrm{Th}(G \cap H) = \{\mathrm{begin}(P) \preceq \mathrm{end}(P)\} .$$

Clearly $\mathrm{create}(A) \preceq \mathrm{end}(P) \in \mathrm{Th}(G) \cap \mathrm{Th}(H) \not\subseteq \mathrm{Th}(G \cap H)$. Note that $\mathrm{create}(A) \preceq \mathrm{end}(P)$ is not even a logical consequence of $\mathrm{Th}(G \cap H)$.

Remark 5.3. The variant of Proposition 5.1 where we replace the theory by its closure (as defined in Section 6) no longer holds as stated. Indeed, we have $\overline{\mathrm{Th}(G \cup H)} = \overline{\mathrm{Th}(G) \cup \mathrm{Th}(G)} \supseteq \overline{\mathrm{Th}(G)} \cup \overline{\mathrm{Th}(H)}$, but the latter containment relation can be strict. To wit, consider G with three artifacts A, B and C and edge $A \to B$, and H with the same three artifacts and $B \to C$. Then $\mathrm{create}(C) \preceq \mathrm{create}(A)$ belongs to $\overline{\mathrm{Th}(G \cup H)}$ but not to $\overline{\mathrm{Th}(G)} \cup \overline{\mathrm{Th}(H)}$.

The variant of Proposition 5.2 with closures does continue to hold as stated. Indeed, we still have $\overline{\mathrm{Th}(G \cap H)} \subseteq \overline{\mathrm{Th}(G)} \cap \overline{\mathrm{Th}(H)}$, where the containment can again be strict. To wit, consider G with three artifacts A, B and C and edges $A \to B \to C$, and consider H which equals G except that identifier B is replaced by a different identifier B'. Then $\mathrm{create}(C) \preceq \mathrm{create}(A)$ belongs to $\overline{\mathrm{Th}(G)} \cap \overline{\mathrm{Th}(H)}$ but not to $\overline{\mathrm{Th}(G \cap H)}$.

5.1. Renaming and merging

By definition, the nodes and roles of an OPM graph are local to the graph. Prior to performing a union or an intersection of two OPM graphs G and H, we may need to resolve some identity issues between the nodes and roles in the graphs [Missier et al. 2010]. For example, some artifact node A in G may represent the same actual artifact as some artifact node B in H[15]. Likewise, role r in edge $P \xrightarrow{r} B$ in G may refer to the same actual role as role s in edge $Q \xrightarrow{s} C$ in H, and also P and Q may represent the same actual process. Moreover, it is equally possible that some node or role is accidentally used in both graphs whereas this node or role does *not* represent the same actual entity across the two graphs. Resolving such identity issues leads to a renaming operation on one or both of the graphs, whereby nodes and roles representing the same actual entity can be renamed to a common node or role; likewise, nodes and roles not representing the same actual entity, but accidentally used in both graphs, can be renamed to distinct nodes or roles.

Definition 5.4 (*Renaming*). Let G and H be OPM graphs, which need not be legal. Let ρ_{Art} be a bijection from Art^G to a finite set Art', let ρ_{Proc} be a bijection from $Proc^G$ to a finite set $Proc'$, and let ρ_{Roles} be a bijection from $Roles^G$ to a finite set $Roles'$, with the sets Art', $Proc'$, and $Roles'$ mutually disjoint. Then H is the *renaming* of G by ρ_{Art}, ρ_{Proc}, and ρ_{Roles}, if the following holds:

— $Art^H = Art'$,
— $Proc^H = Proc'$,
— $Roles^H = Roles'$,
— $GeneratedBy!^H = \{(\rho_{Art}(A), \rho_{Roles}(r), \rho_{Proc}(P)) \mid (A, r, P) \in GeneratedBy!^G\}$;
— $Used!^H = \{(\rho_{Proc}(P), \rho_{Roles}(r), \rho_{Art}(A)) \mid (P, r, A) \in Used!^G\}$;
— $DerivedFrom!^H = \{(\rho_{Art}(A), \rho_{Roles}(r), \rho_{Art}(B)) \mid (A, r, B) \in DerivedFrom!^G\}$;
— $GeneratedBy^H = \left\{(\rho_{Art}(A), \rho_{Proc}(P)) \mid (A, P) \in GeneratedBy^G\right\}$;
— $Used^H = \left\{(\rho_{Proc}(P), \rho_{Art}(A)) \mid (P, A) \in Used^G\right\}$;
— $DerivedFrom^H = \left\{(\rho_{Art}(A), \rho_{Art}(B)) \mid (A, B) \in DerivedFrom^G\right\}$;
— $InformedBy^H = \left\{(\rho_{Proc}(P), \rho_{Proc}(Q)) \mid (P, Q) \in InformedBy^G\right\}$;

Note that Art^G and Art' need not be disjoint; similarly, neither $Proc^G$ and $Proc'$, nor $Roles^G$ and $Roles'$, need to be disjoint. Indeed, ρ_{Art}, ρ_{Proc} and ρ_{Roles} may coincide with

[15]In OPM, node identifiers are scoped to a graph. A node may be associated with a URI by means of a property, allowing it to refer to a Web resource.

the identity function on some of their inputs, i.e., not all artifacts, processes and roles need to be renamed.

Example 5.5. We can rename the graph presented in Figure 8(b) by the following bijections:

— $\rho_{Art}(A) = A$, $\rho_{Art}(C) = B$, $\rho_{Art}(D) = F$, and $\rho_{Art}(E) = E$;
— $\rho_{Proc}(Q) = P$;
— $\rho_{Roles}(s) = r$ and $\rho_{Roles}(s') = r'$.

Then we can take the union of the renamed graph with the graph shown in Figure 8(a), which yields the legal OPM graph presented in Figure 8(c). □

Since the renaming of an OPM graph G is isomorphic to G, and our entire approach (in particular the definition of legality) is "logical" in the sense of Tarski [Tarski 1986], isomorphic graphs have exactly the same properties. Hence, the following is immediate:

PROPOSITION 5.6. *The renaming of a legal OPM graph is legal.*

We next define the following generalization of renaming.

Definition 5.7 (*Merge-renaming*). Let G and H be OPM graphs, which need not be legal. Let ρ_{Art}, ρ_{Proc} and ρ_{Roles} be as in Definition 5.4 except that ρ_{Art}, ρ_{Proc} and ρ_{Roles} need not be bijective: they only need to be surjective (onto) mappings. Then we say that H is the *merge-renaming* of G by ρ_{Art}, ρ_{Proc}, and ρ_{Roles}, exactly if the same equalities of Definition 5.4 hold.

Merge-renaming allows the coalescing of two or more nodes to a single node (or two or more roles to a single role). Coalescing of nodes or roles may be performed when analyzing an OPM graph on a coarser level of detail. But coalescing may also be practical when more information becomes available. For example, in a traffic accident scenario, there may be observations about a "blue car" and other observations about a "Toyota", only to realize later that the blue car is the Toyota.

In contrast to Proposition 5.6, the merge-renaming of a legal OPM graph need not be legal. For example, in Figure 9(c), if we coalesce C and D into a single artifact E, but do not coalesce P and Q, nor their roles, then E has two distinct precise generated-by edges.

As a merge-renaming can coalesce artifacts, such an operation can introduce cycles of derived-from edges to an OPM graph. In the next section, we investigate the consequences of such cycles in OPM graphs.

5.2. Inference of equalities

Our definitions allow the presence of derived-from cycles in legal OPM graphs. By a *derived-from cycle*, we mean a directed simple cycle composed of derived-from edges (precise or imprecise). An OPM graph resulting from a typical experimental provenance collection procedure does not contain such cycles, and indeed the current OPM reference specification forbids them. For example, it would be strange to assert that A is derived from B and that B is derived from A.

Nevertheless, cycles may arise in a graph when, after a merge operation, certain nodes coalesce. Suppose, for example, that we have three artifacts $A \to B \to C$ without a cycle. If an application does not need the full level of detail provided, it may consider, for example, A and C to be the same at a coarser level of detail. As a consequence, a cycle $A \to B \to C = A$ is created.

Thus, we do not want to disallow derived-from cycles in OPM graphs from the outset. It is important, however, to understand the consequences of the presence of such cycles. We observe that they enforce the equality of certain temporal variables. In the preceding example, we would have $\text{create}(A) = \text{create}(B) = \text{create}(C)$.

First of all, we point out that every OPM graph has a trivial model τ_{triv} consisting of a single time-point t_0 with $\tau_{\text{triv}}(u) = t_0$ for every temporal variable u. Indeed, since the temporal theory of an OPM graph consists only of non-strict inequalities, this interpretation trivially satisfies all non-strict inequalities. Of course, that does not mean that this trivial model is the *only* model that the OPM graph possesses. On the contrary: intuitively, on a fine enough temporal granularity, we should expect that every OPM graph indeed possesses a model where all temporal variables can be assigned *distinct* time-points. We observe that this is indeed always possible provided there are no derived-from cycles.

Formally, we fix some OPM graph G for this section. Let us say that a temporal interpretation $(T, \leq, \tau)$ of G is *all-distinct* if $\tau(u) \neq \tau(v)$ for any two distinct temporal variables u and v of G. When, in addition, $\leq$ is a total order on T, we say that τ has the *strict linear order* property.

PROPOSITION 5.8. *If G does not contain any derived-from cycles, then G has an all-distinct temporal model that even satisfies the strict-linear-order property.*

PROOF. We construct a total order on all temporal variables of G that is a temporal model of G under the identity mapping. Since G does not contain any derived-from cycles, we can linearly order all artifacts so that no artifact has a derived-from edge from an artifact coming later in the order. Note that there may be many possibilities for such an ordering. Any such ordering imposes an ordering on the corresponding create-variables. All begin-variables are placed before all create-variables, in some arbitrary order among them, and similarly all end-variables are placed after all create-variables. Finally, a use-variable involving artifact A is placed immediately as a successor of create(A). If there are more than one use-variables for the same artifact A, they can all be placed in an arbitrary order right after create(A). By inspecting the axioms we see that this order satisfies all axioms. □

We note that Proposition 5.8 does not state that all temporal models of cycle-free graphs are all-distinct. Rather, it establishes that one such all-distinct model exists. The next proposition provides a partial converse to Proposition 5.8:

PROPOSITION 5.9. *If G does contain a derived-from cycle of length at least two, then G cannot have an all-distinct temporal model.*

PROOF. Consider a derived-from cycle and let A and B be two distinct artifacts on that cycle. Then any temporal model τ should satisfy $\tau(\text{create}(A)) \leq \tau(\text{create}(B))$ as well as $\tau(\text{create}(B)) \leq \tau(\text{create}(A))$, so $\tau(\text{create}(A))$ equals $\tau(\text{create}(B))$. Hence τ is not all-distinct. □

Propositions 5.8–5.9 do not specify what happens when there are only derived-from cycles of length one (loops). Moreover, for a temporal model τ, they do not specify which distinct variables u and v *cannot* have distinct $\tau(u)$ and $\tau(v)$ if the graph contains derived-from cycles. The next proposition fills these gaps by characterizing exactly when two temporal variables must be equal in all temporal models.

Naturally, for two distinct temporal variables u and v of G, we write $\text{Th}(G) \models u = v$ to denote that both $\text{Th}(G) \models u \preceq v$ and $\text{Th}(G) \models v \preceq u$. Thus, if $\text{Th}(G) \models u = v$, then there is no model of G that is all-distinct since any temporal model τ must satisfy $\tau(u) \leq \tau(v)$ and $\tau(v) \leq \tau(u)$; so $\tau(u) = \tau(v)$. Intuitively, two temporal variables u and v of G such that $\text{Th}(G) \models u = v$ can be seen as indistinguishable in the given temporal model, for example as a result of coalescing some nodes in a graph with a more detailed temporal model.

PROPOSITION 5.10. $\text{Th}(G) \models u = v$ *if and only if u and v match one or more of the following possibilities:*

(1) *u is* create(A)*, v is* create(B)*, and A and B lie together on a derived-from cycle.*

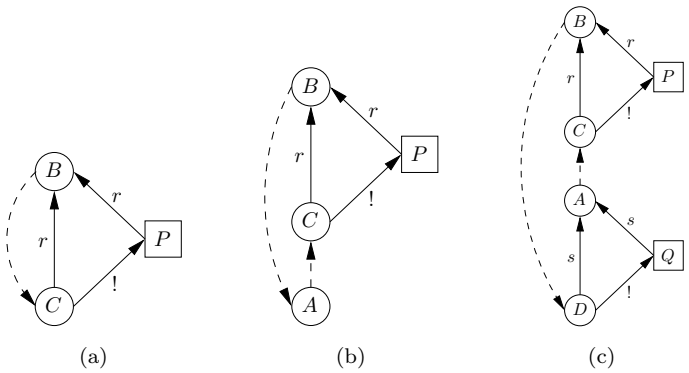

Fig. 9. Graph patterns for Proposition 5.10.

(*2*) *u is* create(B)*, v is* use(P, r, B)*, and in G, the nodes P and B, together with some node C, match the pattern shown in Figure 9(a). Note that B and C need not be distinct.*

(*3*) *u is* create(A)*, v is* use(P, r, B)*, and in G, the nodes P, A and B, together with some node C, match the pattern shown in Figure 9(b). Note that A, B and C need not be distinct.*

(*4*) *u is* use(P, r, B)*, v is* use(Q, s, A)*, and in G, nodes P, Q, A and B, together with some nodes C and D, match the pattern shown in Figure 9(c). Note that A, B, C and D need not be distinct, nor P and Q.*[16]

Proof. The proof of the if-direction amounts to an inspection of involved patterns to verify that $\mathrm{Th}(G) \models u = v$ indeed holds. For example, let us examine pattern in Figure 9(c) in case 4. By Axiom 3 we have $\mathrm{Th}(G) \models \mathrm{create}(B) \preceq \mathrm{use}(P, r, B)$ for edge $P \xrightarrow{r} B$ and $\mathrm{Th}(G) \models \mathrm{create}(A) \preceq \mathrm{use}(Q, s, A)$ for edge $Q \xrightarrow{s} A$. For the triangle (C, B, P, r) and $G \vdash B \dashrightarrow C$, we can apply Rule 7 from Figure 7, so we have $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) \preceq \mathrm{create}(B)$, and thus $\mathrm{Th}(G) \models \mathrm{create}(B) = \mathrm{use}(P, r, B)$. Likewise, for the triangle (D, A, Q, s) and $G \vdash A \dashrightarrow D$, we obtain $\mathrm{Th}(G) \models \mathrm{create}(A) = \mathrm{use}(Q, s, A)$. Since A and B lie on a derived-from cycle, we also have $\mathrm{Th}(G) \models \mathrm{create}(A) = \mathrm{create}(B)$, hence $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) = \mathrm{use}(Q, s, A)$.

The proof of the only-if direction amounts to a lengthy but straightforward inspection of the possible cases where both $\mathrm{Th}(G) \models u \preceq v$ and $\mathrm{Th}(G) \models v \preceq u$, in the characterization of temporal inference provided by Figure 7. For example, in case 4, we clearly see that we can only obtain $\mathrm{Th}(G) \models \mathrm{use}(P, r, B) = \mathrm{use}(Q, s, A)$ by combining the following rules of Figure 7: Rule 9a with again Rule 9a resulting in the pattern from Figure 9(c) with $A = C$ and $B = D$; Rule 9a with Rule 9b resulting in the pattern from Figure 9(c) with $A = C$; and Rule 9b with again Rule 9b resulting in the pattern from Figure 9(c). □

The patterns of Figures 9(a), 9(b), and 9(c) are respectively super graphs of Axiom 8, and cases 7 and 9B in Figure 7. It is interesting to note that a graph that matches Figure 9(c) also matches Figure 9(b) since $B \dashrightarrow A$ can be inferred from $B \dashrightarrow D$ and $D \xrightarrow{s} A$ in Figure 9(c). Likewise, a graph that matches Figure 9(b) also matches Figure 9(a) since $B \dashrightarrow C$ can be inferred from $B \dashrightarrow A$ and $A \dashrightarrow C$ in Figure 9(b).

Hence, by repeated application of Proposition 5.10 (1)–(4), we derive that all use and create time-points for artifacts A, B, C, D in Figure 9(c) are equal. Likewise, we note the

[16]Note that in Figure 9(*c*), if C and D coincide then $C \xrightarrow{!} P$ and $D \xrightarrow{!} Q$ must also coincide for G to be legal.

equality of all use and create time-points for artifacts A, B, C in Figure 9(b) and for B, C in Figure 9(a).

The OPM reference specification does not allow derived-from cycles, but it does not define a merge operation either. This section has demonstrated that a merge operation can introduce derived-from cycles into OPM graphs, but they must satisfy some constraints: all time-points of artifacts involved in a cycle must coalesce to a single time-point. Were a merge operation added to the reference specification, two options are possible. On the one hand, the absence of derived-from cycles can remain a legality constraint, but, therefore, the merge operation should be such that it coalesces all artifacts (and related process) in a cycle and removes all loops to ensure legality is preserved. On the other hand, the constraint on derived-from cycles can be lifted, as it is in this paper, but the OPM edges decorated with time information and involved in cycles should have equal time information whenever the patterns of Proposition 5.10 are matched.

6. REFINEMENT AND COMPLETION

The OPM reference specification introduces the notion of *refinement* as a relation between two graphs: this relation expresses that one graph represents a more complete description of execution than another graph. The term refinement is inspired by the concept of specification refinement in formal methods [Woodcock and Davies 1996]. The concept was only intuitively defined as follows: a graph is a refinement of another if dependencies that can be inferred in the original graph are "preserved" in the refinement. The purpose of this section is to formally ground such a notion of refinement in the context of our temporal semantics.[17]

We fix two legal OPM graphs G and H for use in this section. We also define the following convenient notion.

Definition 6.1. The *logical closure* of a set of inequalities Σ, denoted by $\overline{\Sigma}$, is the set of logical consequences of Σ, i.e., $\overline{\Sigma} = \{\varphi \mid \Sigma \models \varphi\}$.

When context allows, we abbreviate logical closure to closure.

First, we define restriction of an arbitrary set of inequalities Σ to a subset of variables occurring in Σ.

Definition 6.2. Let Σ be a set of inequalities over a set of variables V. Let W be a subset of V. The *restriction* of Σ to W, denoted by $\Sigma|_W$, is the set

$$\Sigma|_W = \{u \preceq v \in \Sigma \mid u, v \in W\} \ .$$

Given our temporal semantics, the intuition of a refinement is the following. Graph H is a refinement of G if all the temporal constraints that can be inferred from $\mathrm{Th}(G)$ can also be inferred from $\mathrm{Th}(H)$. Such definition would be too restrictive, however, as some temporal constraints of $\mathrm{Th}(G)$ may be about temporal variables that do not exist in $\mathrm{Th}(H)$ at all. Indeed, refinements can replace nodes by others (say, when a process is implemented by composing two other processes). Hence, H is a refinement of G, if the temporal constraints that can be inferred from $\mathrm{Th}(G)$ over the common set of variables between H and G, can also be inferred from $\mathrm{Th}(H)$. Formally, the definition is expressed as follows.

Definition 6.3 (*Refinement*). H is a *refinement* of G if

$$\overline{\mathrm{Th}(G)}|_{Vars(G) \cap Vars(H)} \subseteq \overline{\mathrm{Th}(H)} \, .$$

[17] Technically, the OPM reference specification defines refinement between two accounts. However, like graph operations, it is useful to define refinement over OPM graphs. Since accounts correspond to subgraphs, our definition can naturally be used to compare two accounts for refinement as well. We will formally define accounts in Section 7.

In this definition, it is not necessary to restrict $\overline{\mathrm{Th}(H)}$ because $\overline{\mathrm{Th}(H)}$ is on the right-hand side of a set containment. Indeed, for any three sets A, B and C, we have that $(A \cap B) \subseteq C$ iff $(A \cap B) \subseteq (C \cap B)$.

Note that Theorem 4.7 can be used effectively to decide whether a given graph H is a refinement of a given graph G. Still, the definition of refinement is strictly semantic and does not provide much guidance towards constructing a refinement. An interesting open problem is to find a finite set of graph operations that all result in refinements, and such that every refinement can be obtained by using these operations.

Remark 6.4. It is a consequence of our definition that whenever $\mathit{Vars}(G)$ and $\mathit{Vars}(H)$ are disjoint, then H is a refinement of G and vice versa. Indeed, the operation of first adding to G a separate graph component about an entirely disjoint set of nodes, immediately followed by removing the old contents of G, may be considered as an extreme kind of refinement. At any rate our definition of refinements remains open to scientific debate.

6.1. Graph operations and refinement

In this section we investigate the relations between the graph operations defined in Section 5 and refinement. We first investigate the subgraph operation, the union and the intersection. Let G and H be two legal OPM graphs.

PROPOSITION 6.5. *If H is a legal subgraph of G, then G is a refinement of H.*

PROOF. Since H is a subgraph of G, it is clear from Definition 3.6 that $\mathrm{Th}(H) \subseteq \mathrm{Th}(G)$. Hence $\overline{\mathrm{Th}(H)}|_{\mathit{Vars}(H) \cap \mathit{Vars}(G)} = \overline{\mathrm{Th}(H)} \subseteq \overline{\mathrm{Th}(G)}$ as desired. □

Note that a legal subgraph of G is not necessarily a refinement of G. For instance, let G be a use–generate–derive triangle (A, B, P, r), and let H be a subgraph of G consisting of $P \xrightarrow{r} B$ and $A \xrightarrow{!} P$. Then H is legal, but is not a refinement of G: $\mathrm{create}(B) \preceq \mathrm{create}(A) \in \overline{\mathrm{Th}(G)}|_{\mathit{Vars}(G) \cap \mathit{Vars}(H)}$ yet $\mathrm{create}(B) \preceq \mathrm{create}(A) \notin \overline{\mathrm{Th}(H)}$.

The above proposition also applies to the union and intersection, and their operands.

COROLLARY 6.6. *If $G \cup H$ is legal, then $G \cup H$ is a refinement of G.*

Note that G is not necessarily a refinement of a legal $G \cup H$. For example, let G consist of $A \to B$ and node P and let H consist of $A \xrightarrow{!} P$ and node B. So $G \cup H$ is legal and consists of $A \to B$ and $A \xrightarrow{!} P$. Then G is not a refinement of $G \cup H$: $G \cup H \vdash P \to B$, so $\mathrm{create}(B) \preceq \mathrm{end}(P) \in \overline{\mathrm{Th}(G \cup H)}|_{\mathit{Vars}(G \cup H) \cap \mathit{Vars}(G)}$ yet $\mathrm{create}(B) \preceq \mathrm{end}(P) \notin \overline{\mathrm{Th}(G)}$.

However, if G and H are node-disjoint, then G is a refinement of $G \cup H$.

COROLLARY 6.7. *G is a refinement of $G \cap H$.*

Note that $G \cap H$ is not necessarily a refinement of G. For example, let G consists of $A \to Q$, $A \xrightarrow{!} P$ and $P \to Q$, and let H consist of $A \to Q$ and $P \to Q$. Then $G \cap H$ equals H, which is not a refinement of G: $\mathrm{create}(A) \preceq \mathrm{end}(P) \in \overline{\mathrm{Th}(G)}|_{\mathit{Vars}(G) \cap \mathit{Vars}(H)}$, yet $\mathrm{create}(A) \preceq \mathrm{end}(P) \notin \overline{\mathrm{Th}(G \cap H)}$.

Next we investigate the merge-renaming operation. Let H be a renaming of G by ρ_{Art}, ρ_{Proc} and ρ_{Roles} (conforming to Definition 5.4). Then G is not necessarily a refinement of H, or vice versa. For instance, let G consist of $A \to B$, let $\rho_{\mathit{Art}}(A) = B$ and $\rho_{\mathit{Art}}(B) = A$, then H consists of $B \to A$. They are clearly not each others refinements.

Still, if one needs to apply a merge-renaming on a graph before performing a union or an intersection, a merge-renaming operation that preserves the temporal constraints of the original would be advisable. This motivates the following restricted version of the merge-renaming operation:

Definition 6.8 (*Proper merge-renaming*). Let ρ_{Art}, ρ_{Proc} and ρ_{Roles} be as in Definition 5.7, and let ρ be the point-wise union of ρ_{Art}, ρ_{Proc} and ρ_{Roles}. Then we use $\rho(G)$ to denote the merge-renaming of G by ρ_{Art}, ρ_{Proc} and ρ_{Roles}.

We call a merge-renaming $\rho(G)$ *proper* if for any node (or role) x in G, if $\rho(x) \neq x$ but $\rho(x)$ is also a node (or role) in G, then $\rho(\rho(x)) = \rho(x)$.

Intuitively, in a proper merge-renaming operation, some nodes/roles are preserved, whereas all the others are either renamed into new ones or coalesced into preserved ones. Such an operation disallows arbitrary renaming and permutation of nodes/roles. Although it seems overly restrictive, it makes sense in the OPM model, because nodes and roles are actually *identifiers*. Hence, coalescing of nodes/roles, or renaming them into new ones, is a form of identity resolution. For example, witnesses of a car accident may speak of a "blue car" or a "Toyota". Later on, one realizes the witnesses talked about the same car, so both "blue car" and "Toyota" should be renamed to the car's registration number (while "blue car" and "Toyota" become its annotations). Likewise, hospitals frequently admit unconscious patients under a temporary ID. Later on, they are either matched to an already known patient or to a new ID if the patient is admitted for the first time.

We conclude this section by showing that a proper and legal merge-renaming of a graph is indeed a refinement of the original graph.

THEOREM 6.9. *Let G be a legal OPM graph, and let $\rho(G)$ be a proper and legal merge-renaming of G, for some ρ. Then $\rho(G)$ is a refinement of G.*

To prove the theorem we use the following auxiliary lemmas.

LEMMA 6.10. *Let G be a legal OPM graph, and let $\rho(G)$ be a legal merge-renaming of G, for some ρ. Then if $G \vdash X \dashrightarrow Y$, also $\rho(G) \vdash \rho(X) \dashrightarrow \rho(Y)$, i.e., a legal merge-renaming preserves edge-inference.*

The above lemma is readily verified.

LEMMA 6.11. *Let G be a legal OPM graph, and let $\rho(G)$ be a proper merge-renaming of G, for some ρ. If a node (or role) x belongs to both G and to the image of ρ, then $\rho(x) = x$.*

Indeed, if x is in the image of ρ, then there exist a node (role) y in G, such that $\rho(x) = y$. If $x = y$, then $\rho(x) = x$ holds immediately. If $x \neq y$, then $\rho(y) \neq y$, so $\rho(\rho(y)) = \rho(y)$ (by Definition 6.8). Thus $\rho(x) = x$ holds as desired.

We can now prove the theorem. We need to prove the following:

$$\overline{\mathrm{Th}(G)}|_{Vars(G) \cap Vars(\rho(G))} \subseteq \overline{\mathrm{Th}(\rho(G))}\,.$$

Let $Commons = Vars(G) \cap Vars(\rho(G))$. For each inequality $\varphi \in \overline{\mathrm{Th}(G)}|_{Commons}$, we must show that φ belongs to $\overline{\mathrm{Th}(\rho(G))}$. We know from Theorem 4.7, illustrated in Figure 7, that each such inequality is associated with a pattern in G. Therefore, we need to find a similar pattern in $\rho(G)$ that produces *exactly* the same inequality.

Let $\varphi \in \overline{\mathrm{Th}(G)}|_{Commons}$. We follow the axioms and rules presented in Figure 7, the axioms first, to cover all possible forms that φ may assume:

(a) φ is $\mathrm{begin}(P) \preceq \mathrm{end}(P)$, for some P in G. Since $\mathrm{begin}(P), \mathrm{end}(P) \in Commons$, P is also present in $\rho(G)$. By Axiom 1, $\mathrm{begin}(P) \preceq \mathrm{end}(P) \in \overline{\mathrm{Th}(\rho(G))}$.

(b) φ is $\mathrm{begin}(P) \preceq \mathrm{create}(A)$ or $\mathrm{create}(A) \preceq \mathrm{end}(P)$, for some $A \xrightarrow{!} P$ in G. Since $\mathrm{create}(A), \mathrm{begin}(P), \mathrm{end}(P) \in Commons$, we also have A and P in $\rho(G)$. From $A \xrightarrow{!} P$ in G, by Definition 6.8, we have $\rho(A) \xrightarrow{!} \rho(P)$ in $\rho(G)$. We can apply Lemma 6.11 to A and P, obtaining $\rho(A) = A$ and $\rho(P) = P$. Hence, $A \xrightarrow{!} P$ is also in $\rho(G)$ and,

by Axiom 2, both $\text{begin}(P) \preceq \text{create}(A)$ and $\text{create}(A) \preceq \text{end}(P)$ belong to $\overline{\text{Th}(\rho(G))}$. Note that $A \xrightarrow{!} P$ in G and $A \xrightarrow{!} P$ in $\rho(G)$ need not have the same role.

(c) φ is one of the following: $\text{begin}(P) \preceq \text{use}(P, r, A)$, $\text{use}(P, r, A) \preceq \text{end}(P)$, or $\text{create}(A) \preceq \text{use}(P, r, A)$, for some $P \xrightarrow{r} A$ in G. The variable $\text{use}(P, r, A)$ belongs to *Commons*; that is only possible if edge $P \xrightarrow{r} A$ is also present in $\rho(G)$. By Axiom 3, $\text{begin}(P) \preceq \text{use}(P, r, A)$, $\text{use}(P, r, A) \preceq \text{end}(P)$, and $\text{create}(A) \preceq \text{use}(P, r, A)$ belong to $\overline{\text{Th}(\rho(G))}$.

(d) φ is $\text{use}(P, r, B) \preceq \text{create}(A)$, for some $G \triangle (A, B, P, r)$. For $P \xrightarrow{r} B$ in G we apply case c, so $P \xrightarrow{r} B$ is also in $\rho(G)$. For $A \xrightarrow{!} P$ in G, by case b, $A \xrightarrow{!} P$ belongs to $\rho(G)$. We can apply Lemma 6.11 to A, B, and r, obtaining $\rho(A) = A$, $\rho(B) = B$, and $\rho(r) = r$. For $A \xrightarrow{r} B$ in G, we apply Definition 6.8, so $\rho(A) \xrightarrow{\rho(r)} \rho(B)$ in $\rho(G)$, and thus $A \xrightarrow{r} B$ in $\rho(G)$. Clearly, $\rho(G) \triangle (A, B, P, r)$, hence $\text{use}(P, r, B) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$ (Axiom 8).

(e) φ is $\text{create}(B) \preceq \text{create}(A)$, for $G \vdash A \dashrightarrow B$. Since $\text{create}(A)$, $\text{create}(B) \in \mathit{Commons}$, A and B also belong to $\rho(G)$. From $G \vdash A \dashrightarrow B$, by Lemma 6.10, we have $\rho(G) \vdash \rho(A) \dashrightarrow \rho(B)$. We can apply Lemma 6.11 to A and B, obtaining $\rho(A) = A$ and $\rho(B) = B$. Hence $\rho(G) \vdash A \dashrightarrow B$ as desired and, by Rule 1, $\text{create}(B) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$.

(f) φ is $\text{begin}(P) \preceq \text{create}(A)$, for $G \vdash A \dashrightarrow P$. Since $\text{create}(A)$, $\text{begin}(P)$, $\text{end}(P) \in \mathit{Commons}$, A and P belong to $\rho(G)$. From $G \vdash A \dashrightarrow P$, by Lemma 6.10, $\rho(G) \vdash \rho(A) \dashrightarrow \rho(P)$. We can apply Lemma 6.11 to A and P, obtaining $\rho(A) = A$ and $\rho(P) = P$. Thus $\rho(G) \vdash A \dashrightarrow P$ and, by Rule 2, $\text{begin}(P) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$.

(g) φ is $\text{create}(A) \preceq \text{end}(P)$, for $G \vdash P \dashrightarrow A$. Since $\text{create}(A)$, $\text{begin}(P)$, $\text{end}(P) \in \mathit{Commons}$, A and P belong to $\rho(G)$. From $G \vdash P \dashrightarrow A$, by Lemma 6.10, we have $\rho(G) \vdash \rho(P) \dashrightarrow \rho(A)$. We can apply Lemma 6.11 to A and P, obtaining $\rho(A) = A$ and $\rho(P) = P$. Therefore, $\rho(G) \vdash P \dashrightarrow A$ and, by Rule 3, $\text{create}(A) \preceq \text{end}(P) \in \overline{\text{Th}(\rho(G))}$.

(h) φ is $\text{begin}(Q) \preceq \text{end}(P)$, for $G \vdash P \dashrightarrow Q$. Since $\text{begin}(P)$, $\text{end}(P)$, $\text{begin}(Q)$, $\text{end}(Q) \in \mathit{Commons}$, P and Q belong to $\rho(G)$. From $G \vdash P \dashrightarrow Q$, by Lemma 6.10, $\rho(G) \vdash \rho(P) \dashrightarrow \rho(Q)$. We can apply Lemma 6.11 to P and Q, obtaining $\rho(P) = P$ and $\rho(Q) = Q$. Hence $\rho(G) \vdash P \dashrightarrow Q$ and, by Rule 4, $\text{begin}(Q) \preceq \text{end}(P) \in \overline{\text{Th}(\rho(G))}$.

(i) φ is $\text{create}(B) \preceq \text{use}(P, r, A)$, for $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$. By applying cases c and e to $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$, respectively, we have $P \xrightarrow{r} A$ in $\rho(G)$ and $\rho(G) \vdash A \dashrightarrow B$. By Rule 5, $\text{create}(B) \preceq \text{use}(P, r, A) \in \overline{\text{Th}(\rho(G))}$.

(j) φ is $\text{begin}(Q) \preceq \text{use}(P, r, A)$, for $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$. By applying cases c and f to $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$, respectively, we have $P \xrightarrow{r} A$ in $\rho(G)$ and $\rho(G) \vdash A \dashrightarrow Q$. By Rule 6, $\text{begin}(Q) \preceq \text{use}(P, r, A) \in \overline{\text{Th}(\rho(G))}$.

(k) φ is $\text{use}(P, r, C) \preceq \text{create}(A)$, for $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$. By applying cases d and e to $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$, respectively, we have $\rho(G) \triangle (B, C, P, r)$ and $\rho(G) \vdash A \dashrightarrow B$. By Rule 7, $\text{use}(P, r, C) \preceq \text{create}(A) \in \overline{\text{Th}(\rho(G))}$.

(l) φ is $\text{use}(P, r, B) \preceq \text{end}(Q)$, for $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$. By applying cases d and g to $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$, respectively, we have $\rho(G) \triangle (A, B, P, r)$ and $\rho(G) \vdash Q \dashrightarrow A$. By Rule 8, $\text{use}(P, r, B) \preceq \text{end}(Q) \in \overline{\text{Th}(\rho(G))}$.

(m) φ is $\text{use}(P, r, B) \preceq \text{use}(Q, s, A)$, for $G \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$ in G, and either $A = C$ or $G \vdash A \dashrightarrow C$. By applying cases d and c to $G \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$, respectively, we obtain $\rho(G) \triangle (C, B, P, r)$ and $Q \xrightarrow{s} A$ in $\rho(G)$. If $A = C$, then A clearly belongs to G and to the image of ρ, so, by Lemma 6.11, $\rho(A) = A$. If $G \vdash A \dashrightarrow C$, then, by case e, $\rho(G) \vdash A \dashrightarrow C$. We can thus apply either Rule 9a or Rule 9b, so $\text{use}(P, r, B) \preceq \text{use}(Q, s, A) \in \overline{\text{Th}(\rho(G))}$.

6.2. Completion Operations

The edge-inference rules introduced in Definition 4.5 allow new edges to be inferred in an OPM graph, the nodes of the graph remaining unchanged. The OPM reference specification also defines *completion operations*, which add new nodes and new edges to a graph. The addition of new nodes is an object-creating operation which, in logic programming, is implemented using Skolem functions [Hull and Yoshikawa 1990].

The rationale for such completion operations was to provide syntactic transformations over graphs, which offer an *explanation* for some OPM edges. First, we present the operations graphically and intuitively, before formalizing them. According to Figure 10(a), *process introduction* states that if an artifact B was derived from an artifact A, then there exist a process R and role r such that B was generated by R with role r and R used A. The operation does not specify which process was involved, nor the role of B with regard to this process, but it states that such a process R and role r existed. Likewise, *artifact introduction*, presented in Figure 10(b), states that if there is a process Q that was informed by a process P, then there exist an artifact C and a role s such that Q used C precisely with role s and C was generated by P. Again, the completion does not specify which artifact and role were involved.[18]

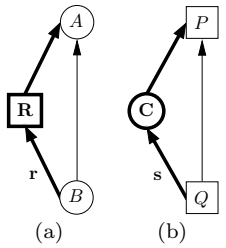

Fig. 10. Completion operations: (a) process introduction and (b) artifact introduction.

We can see that process introduction generalizes the use–generate–derive triangle of Figure 3 to imprecise derived-from and used-edges. We refer to this triangle as an *imprecise use–generate–derive triangle*. Likewise, artifact introduction recognizes the existence of a complementary *imprecise use–generate–inform triangle*. (We note that there is no precise use–generate–inform triangle, since informed-by edges are always imprecise.)

Completion operations should be considered in conjunction with legality constraints. For instance, Figure 11(a) depicts an artifact A_2 derived from two artifacts A_0 and A_1. The process introduction operation can be applied twice here, but the legality constraint requires an artifact to be generated by a single process. Hence, Figure 11(b) displays the *only possible* completion, where introduced process P used both A_0 and A_1, and generated A_2.

Application of process insertion to the graph of Figure 12(a) entails that there is a process that used A_0 and that generated A_2 precisely, but the legality constraint implies that this process is P. Hence, we can derive that P used A_0, which is the inference of a used-edge as in Definition 4.5.

Completion operations can introduce new nodes in a graph, but can result in some uncertainty as illustrated by Figures 13 and 14. In Figure 13, it is unknown whether the two processes P_0 and P_1 introduced by the process introduction operation are identical. Likewise, in Figure 14, it is not known whether the two artifacts A_0 and A_1 introduced by artifact introduction are the same. This uncertainty is formalized below to the effect

[18]The reference specification also defines a third completion operation called artifact elimination, but that operation is nothing else than the inference of informed-by edges.

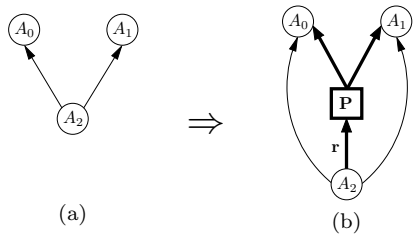

Fig. 11. Process introduction producing a legal graph.

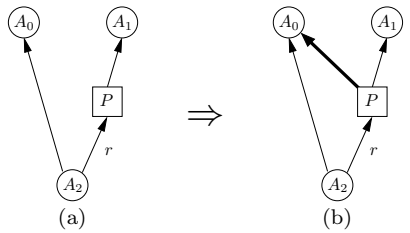

Fig. 12. When completion becomes edge inference.

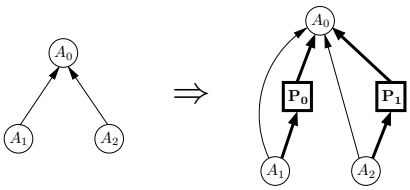

Fig. 13. Uncertainty: is $P_0 = P_1$ or $P_0 \neq P_1$?

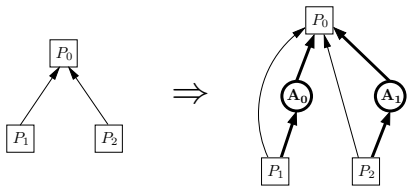

Fig. 14. Uncertainty: is $A_0 = A_1$ or $A_0 \neq A_1$?

that completion is a non-deterministic operation: a given graph may have several possible completions.

To define completion formally, we first formalize the notions of triangle relevant to completion operations.

Definition 6.12 (*Complete Triangle for* $B \to A$). Let G be an OPM graph with two artifacts A and B and an imprecise edge $B \to A$. Graph G contains a complete triangle for $B \to A$ if there exists a process R and role r in G, with edges $B \xrightarrow{r} R$ and $R \to A$. We then say that B, A, R, r constitute an imprecise use–generate–derive triangle.

Definition 6.13 (*Complete Triangle for* $Q \to P$). Let G be an OPM graph with two processes Q and P and an informed-by edge $Q \to P$. Graph G contains a complete triangle for $Q \to P$ if there exists an artifact C and role s in G, with edges $Q \xrightarrow{s} C$ and $C \to P$. We then say that Q, P, C, s constitute an imprecise use–generate–inform triangle.

There can be at most one complete triangle for a given imprecise edge $B \to A$ in a legal OPM graph G, since B can only be generated by one process R. On the other hand, there may be several complete triangles for a given imprecise edge $Q \to P$: for instance, two artifacts C_1, C_2 could be generated by P and used by Q, with respective roles s_1 and s_2.

We then define a completion operation as possibly introducing a node, a role, and edges, as appropriate, in order to form complete triangles.

Definition 6.14 (*CompletionOperation*). Let G be an OPM graph. A graph H results from a completion operation on G, if H was obtained as follows:

— H is the result of *process introduction* for $B \to A$ in G, with $A, B \in Art^G$, if there exists a process R and a role r, such that:
  — $Proc^H = Proc^G \cup \{R\}$,
  — $Roles^H = Roles^G \cup \{r\}$,
  — $GeneratedBy!^H = GeneratedBy!^G \cup \{(B, r, R)\}$,
  — $Used^H = Used^G \cup \{(R, A)\}$,
  — all other sets remain the same;

— H is a result of *artifact introduction* for $Q \to P$ in G, with $P, Q \in Proc^G$, if there exists an artifact C and a role s, such that:
  — $Art^H = Art^G \cup \{C\}$,
  — $Roles^H = Roles^G \cup \{s\}$,
  — $GeneratedBy^H = GeneratedBy^G \cup \{(C, P)\}$,
  — $Used!^H = Used!^G \cup \{(Q, s, C)\}$,
  — all other sets remain the same.

We note that a completion operation can introduce new nodes R and C or reuse existing ones in G. Likewise, they can create new edges or reuse existing ones. In some cases, a completion operation is exactly an edge inference (cf. inference of used-edges in Definition 4.5).

Not all completion operations lead to legal graphs. Hence, a completion operation applied to legal graph G is said to be *valid* if it results in a legal graph H. We then call H a *valid completion* of G.

Since G is always a subgraph of H, the result of a completion is always a refinement of G (Proposition 6.5). The converse does not hold, however. Indeed, let us consider a graph G with imprecise edges $C \to A$ and $C \to B$, and an isolated process P. If graph H is the result of completing $C \to A$ via P, we can infer $H \vdash P \dashrightarrow B$. Hence, the inequality $\text{create}(B) \preceq \text{end}(P)$ holds in H but not in G. Alternatively, let us consider a graph G with imprecise edges $Q \to P$ and $B \to A$. If graph H is the completion of $Q \to P$ via B, then $H \vdash Q \dashrightarrow A$, which cannot be inferred in G.

So, in summary, graphs can be completed non-deterministically, but completions are a refinement of the original graph.

7. MULTI-ACCOUNT OPM GRAPH

In the OPM reference specification [Moreau et al. 2011], OPM graphs can contain multiple *accounts*. Accounts are used to identify parts of a large, integrated, OPM graph, in which each account is a coherent OPM graph in itself. An account should be perceived as one perspective on what happened during a past execution.

Example 7.1. Recall the graph of Figure 1 describing the ordering of an e-book and a toy from an e-shop. Assume the e-book is for Alice and the toy is for Alice's young son Bob. Figure 15 shows an extension of the graph with a second account that accounts for Bob's simple perspective of his mother getting him a new toy. Bob's account is drawn in another color. Note that the nodes 'order' and 'toy' belong to both accounts, as well as the edge between them. □

We define a multi-account OPM graph as follows:

Definition 7.2 (*Multi-account OPM graph*). A multi-account OPM graph is a structure

$$(Art, Proc, Roles, GeneratedBy!, Used!, DerivedFrom!, GeneratedBy, Used, DerivedFrom, InformedBy, accountOf)$$

where

— Art and $Proc$ are two disjoint finite sets of elements called *artifacts* and *processes*, respectively;
— $Roles$ is a finite set of elements called *roles*;
— $GeneratedBy! \subseteq Art \times Roles \times Proc$;
— $Used! \subseteq Proc \times Roles \times Art$;
— $DerivedFrom! \subseteq Art \times Roles \times Art$;
— $GeneratedBy \subseteq Art \times Proc$;
— $Used \subseteq Proc \times Art$;

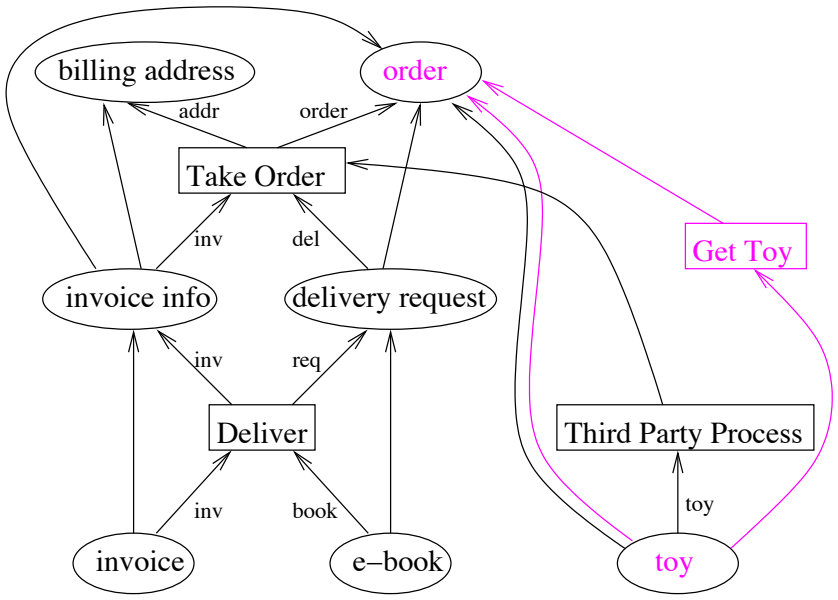

Fig. 15. OPM graph with two accounts.

— $DerivedFrom \subseteq Art \times Art$;
— $InformedBy \subseteq Proc \times Proc$;
— $accountOf$ is a function from $Nodes \cup Edges$ to $\mathcal{P}(Account)$, the set of all subsets of $Account$, where
  — $Account$ is a finite set of elements called *accounts*,
  — $Nodes = Art \cup Proc$, and
  — $Edges = GeneratedBy! \cup Used! \cup DerivedFrom! \cup GeneratedBy \cup Used \cup DerivedFrom \cup InformedBy$.

The sets Art, $Proc$, $Roles$ and $Account$ are mutually disjoint.

In a multi-account OPM graph G, every node and edge of G can be associated with zero, one, or more accounts, which is captured by the function $accountOf$, provided as the last constituent of graph G.

For a given OPM graph G, for any $A \in Art$, we call $accountOf(A)$ the *account membership* of A in G. Likewise, for any $P \in Proc$, we call $accountOf(P)$ the account membership of P in G.

For a precise edge e in G, of the form (x, r, y), or for an imprecise edge e in G, of the form (x, y), we say that x and y are *incident* to e, and denote this by the predicates $isIncident(x, e)$ and $isIncident(y, e)$.

To ensure that the source and destination of an edge belong to the same account as the edge, a node "inherits" and cumulates the account memberships of the edges it is incident to. Formally, this is expressed as follows.

Definition 7.3 (*Effective Account*). For a given OPM graph G we define the *effective-account function*

$$effectiveAccountOf^G : Nodes \cup Edges \to \mathcal{P}(Account)$$

as follows:

— If $x \in Nodes$, then

$$effectiveAccountOf^G(x) = accountOf(x) \cup \bigcup \{accountOf(e) \mid e \in Edges \text{ and } isIncident(x, e)\}.$$

— If $x \in Edges$, then

$$effectiveAccountOf^G(x) = accountOf(x).$$

An *account view* is the single-account OPM graph extracted from a multi-account OPM graph by restricting attention to the nodes and edges associated with a given account. Formally, this operation is defined as follows.

Definition 7.4 (*Account View*). For a given multi-account OPM graph G and an account α, we define the *account view of* G *according to* α, denoted by $view(G, \alpha)$, as follows:

— $Art^{view(G,\alpha)} = \left\{A \in Art \mid \alpha \in effectiveAccountOf^G(A)\right\}$;
— $Proc^{view(G,\alpha)} = \left\{P \in Proc \mid \alpha \in effectiveAccountOf^G(P)\right\}$;
— $Roles^{view(G,\alpha)} = \{r \in Roles \mid (x, r, y) \text{ is an edge in } view(G, \alpha)\}$;
— $GeneratedBy!^{view(G,\alpha)} = \left\{(A, r, P) \mid \alpha \in effectiveAccountOf^G(A, r, P)\right\}$;
— $Used!^{view(G,\alpha)} = \left\{(P, r, A) \mid \alpha \in effectiveAccountOf^G(P, r, A)\right\}$;
— $DerivedFrom!^{view(G,\alpha)} = \left\{(A, r, B) \mid \alpha \in effectiveAccountOf^G(A, r, B)\right\}$;
— $GeneratedBy^{view(G,\alpha)} = \left\{(A, P) \mid \alpha \in effectiveAccountOf^G(A, P)\right\}$;
— $Used^{view(G,\alpha)} = \left\{(P, A) \mid \alpha \in effectiveAccountOf^G(P, A)\right\}$;
— $DerivedFrom^{view(G,\alpha)} = \left\{(A, B) \mid \alpha \in effectiveAccountOf^G(A, B)\right\}$;
— $InformedBy^{view(G,\alpha)} = \left\{(P, Q) \mid \alpha \in effectiveAccountOf^G(P, Q)\right\}$.

Definition 7.5. A multi-account OPM graph is called *legal* if all its account views are legal.

Note that in a legal multi-account OPM graph, we can perform temporal inferences on each account view, although separately. It is difficult to define temporal inference on the whole graph, since each account potentially provides a different perspective. However, one can perform various graph operations to combine different accounts into a single one, and then perform temporal inference on the latter. One can also establish whether one account is a refinement of another one.

8. RELATED WORK

With well over 400 publications [Moreau 2010b] on the topic of provenance, this section focuses on OPM-specific related work. For a broader perspective, we refer the reader to comprehensive surveys for provenance and e-science [Simmhan et al. 2005], provenance and databases [Buneman et al. 2008; Cheney et al. 2009] and provenance and the Web [Moreau 2010b].

Cheney [Cheney 2010] investigates the use of structural causal models as a semantics for provenance graphs, and relates some OPM concepts to notions of actual cause and explanation proposed by Halpern and Pearl [Halpern and Pearl 2005]. Cheney considers only a small fragment of OPM, having only used and generated-by edges, and interpreting processes as functions. As a consequence he considers some inferences, like those considered

in Remark 2.4, which are not part of the OPM reference specification and which are unsound for the temporal semantics considered in this paper. On the other hand, Cheney's semantics attempts the more ambitious goal of providing a global approximation (using the predictive nature of causal models) for the program being executed (without having its explicit code), so that its behavior can be repeated for any arbitrary input; our semantics is silent about the predictive nature of OPM graphs.

Missier and Goble [Missier and Goble 2011] address the question of whether, for any OPM graph, there exists a plausible workflow in the Taverna workflow language, which could have generated the graph. To this end, they identify the extra information that should be captured as part of an OPM graph so that the mapping from OPM to a workflow representation can be derived. Thus, this work derives an executable semantics for OPM, obtained by composing their translation and the Taverna semantics. It however does not tackle OPM in full, ignoring accounts, time and refinement; their translation should be revisited to leverage the distinction between precise and imprecise edges, introduced in this paper. Similarly, Kwasnikowska and Van den Bussche [Kwasnikowska and Van den Bussche 2008] map the NRC data flow model, a formally specified data model for workflows, to OPM.

Moreau [Moreau 2010a] proposes the reproducibility semantics for OPM, which is a denotational semantics characterizing how an OPM graph can be used to reproduce a past computation; a "re-execution" of such a graph results in a new OPM graph, and mapping of nodes from the original graph to the new graph. By doing so, he identifies a class of reproducible OPM graphs. The reproducibility semantics assumes a mapping of each process to a function (taking some inputs artifacts, and generating some output artifact), and, like Cheney's causal semantics of OPM, sees an OPM graph as a function operating on inputs and producing outputs. Moreau defines a notion of refinement, corresponding to the nested execution of procedures. Future work could try to integrate Moreau's reproducibility semantics and the temporal semantics presented in this paper.

Several approaches are specializing OPM to specific application domains or facets of computing. Groth and Moreau introduce the D-Profile [Groth and Moreau 2011], as a specialization of OPM for distributed systems. Their profile comprises artifact and process types and graph patterns to describe communications in distributed systems. Similarly, Freitas et al. introduce types of processes, artifacts and agents to describe data publication over the Web [Freitas et al. 2011].

Several teams have adopted OPM and implemented inferences, as prescribed by the reference specification. To this end, many teams have exploited Semantic Web technologies, such as OWL and SWRL, to implement OPM reasoning: e.g., Tupelo [McGrath and Futrelle 2008], OPMO [Moreau et al. 2010], Provenance Challenge 3 Tetherless [Ding et al. 2011], or, reproducibility service [Moreau 2010a]; alternatively, some teams used recursive queries in relational databases: e.g., OPMProv [Lim et al. 2011]. It is an open question as to whether the complete specification of PROV-CONSTRAINTS can be implemented in any of the OWL2 profiles.

W3C standardization activies. As part of the W3C Provenance Incubator activity [W3C Provenance Incubator Activity 2010], mappings of multiple provenance ontologies to OPM were defined [Sahoo et al. 2010]. These mappings showed that concepts such as processes and artifacts mapped quite naturally between models. The mappings did not take into account the temporal meaning of the various data models; revisiting these mappings in the light of this temporal semantics would provide a better correspondence between ontologies. Likewise, Miles explains how Dublin Core provenance-related concepts can be translated into OPM graphs [Miles 2011]. Given that Dublin Core also introduces time, a finer-grained mapping could be derived, based on the temporal semantics introduced in this paper.

The aim of the W3C Provenance Working Group [W3C PROV 2011] was to support the widespread publication and use of provenance information of Web documents, data, and resources. The Working Group has developed PROV, a standard language for exchanging provenance information among applications [Moreau et al. 2013]. As mentioned in the Introduction, the PROV data model has been influenced by the OPM data model. The PROV data model has many more features but we restrict attention in the following discussion on how it can represent the features considered in the present paper.

In PROV, artifacts and processes are called "entities" and "activities", respectively. Derived-from, generated-by, used, and informed-by edges are stated in PROV in the form of facts with predicates 'wasDerivedFrom', 'wasGeneratedBy', 'used', and 'wasInformedBy', respectively. Each fact has an identifier, which can be unnamed, in which case an existential variable is used as identifier. The identifiers occurring in wasGeneratedBy and used facts serve as temporal variables for the corresponding timepoints of creation of an artifact (entity) and usage of an artifact by a process (activity), respectively. The timepoints of beginning and ending of a process can be indicated in PROV by the identifiers occurring in wasStartedBy and wasEndedBy facts. Use-generate-derive triangles can be stated in PROV by referring, in the wasDerivedFrom fact, to the identifiers occuring in the used and wasGeneratedBy facts. Such wasDerivedFrom facts can then be considered as representing precise derived-from edges.

PROV facts may carry temporal annotations; in this way, a specific (possibly partial) temporal interpretation may be provided. The temporal annotations are subject to *constraints* in the form of inequalities [Cheney et al. 2013]. This allows us to compare the axioms from Definition 3.6 with the PROV constraints.

(1) Axiom 1 is present in PROV as Constraint 30.
(2) Axiom 2 is present in PROV as Constraint 34.
(3) Axiom 3 is present in PROV as Constraints 33 and 37 combined.
(4) Axiom 4 is present in PROV as Constraint 42.[19]
(5) Axiom 5 is not present, as imprecise generated-by edges cannot be directly asserted as such in PROV.
(6) Axiom 6 is not present, as imprecise used edges cannot be directly asserted as such in PROV.[20]
(7) Axiom 7 is present in PROV as Constraint 35.
(8) Axiom 8, finally, is present in PROV as Constraint 41.

The Working Group [Moreau et al. 2013] is further defining two relations, alternateOf and specializationOf, which in the context of OPM would link two artifacts that denote a same thing in the world. If such relations are asserted, then the artifacts that are linked by such relations are candidate for the merge operation defined in this paper.

[19] With the caveat that PROV specifies strict inequalities.

[20] Here, however, one may simulate an imprecise used edge $p \to e$ (with p a process/activity and e an artifact/entity) with a used-fact with an unnamed identifier. The usage event will then be represented by an existential variable, which we denote by `_u` here:

```
used(_u; p, e, -)
wasEndedBy(end; p, -, -, -)
wasGeneratedBy(gen; e, -, -)
```

The identifiers `end` and `gen` serve to represent the temporal variables create(e) and end(p), respectively. Now PROV Constraint 37 yields that `gen` must precede `_u`, and Constraint 33(2) yields that `_u` must precede `end`. Since `_u` is an existential variable, we can express this as

$$\exists u : gen < u < end$$

which amounts to create(e) < end(p) which is what Axiom 6 would produce.

9. CONCLUSION

In this paper we have shown how OPM can be given a formal semantics. Let us summarize again the main steps of our approach. We have first syntactically enriched OPM by making an explicit distinction between precise and imprecise edges. We have then defined a set of temporal variables associated to an OPM graph. These variables represent the timepoints of creation of artifacts, of beginning and ending of processes, and of usage of artifacts by processes. An OPM graph is then viewed as a logical theory, i.e., a set of formulas. These formulas are simple inequalities, and are produced from the edges in the graph, in the way specified in Definition 3.6. Now any way to assigning concrete timepoint values to the different events in a graph is called an interpretation of the graph (Definition 3.5). These values may in principle come from any partially ordered set, although typically this partially ordered set will be the integers or the real or rational numbers. Only when the interpretation satisfies the inequalities inherent to the graph, i.e., the inequalities from its theory, is the interpretation considered to be satisfactory; we then call it a temporal model of the graph (Definition 3.8).

A very natural question now is, given an OPM graph, to understand which inequalities are guaranteed to be satisfied in every of its temporal models. Our main Theorem 4.7 shows that these "implied" inequalities allow a very clear and concise characterization: to verify whether an inequality is implied, it suffices to match its associated graphical pattern against the graph. The comprehensive list of graphical patterns is given in Figure 7. Crucially, these graphical patterns involve the notion of "inferred edges". These are edges that can be traced out in the graph by following paths on derived-from edges, as pictorially illustrated in Figure 4 and formally defined in Definition 4.5. Edge inference was already considered in the original OPM reference specification, but purely as a syntactical game, without any semantical justification. Moreover, we had to enrich edge inference to account better for precise edges. In fairness, however, it turns out that, when use-timepoints are ignored, the original OPM edge inference rules have been complete all along (Section 4.3). This result comes full circle and is very satisfying, as it confirms formally that the original intuition behind OPM (which was developed as a community effort) was largely correct.

The computational complexity of deciding whether a given inequality belongs to the temporal theory of a given OPM graph is polynomial-time. The complexity is actually dominated by the transitive closure computation on derived-from edges that is needed to perform the edge inferences given in Figure 4. After that, inference amounts to retrieving the relevant graph patterns listed in Figure 7. Each graph pattern can be implemented by a fixed conjunctive database query applied to the graph augmented with inferred edges. Hence, techniques for recursive and join query processing, well established in database systems, can be applied for inferencing in large provenance graphs.

Our semantics is explicitly temporal in nature; the time aspect of OPM was kept in the core specification[21] since it is regarded as fundamental to the model. In this paper, we have shown that time is fundamental to the core of OPM.

OPM is "merely" an exchange format for representing provenance information; the next question then, of course, is *how* to extract provenance information out of running systems, or out of databases. Much research exists on this question, and we refer to the surveys [Simmhan et al. 2005; Buneman et al. 2008; Cheney et al. 2009] for more information.

Further research. Some open problems within the scope of the present paper that we have left open are the following. First, the OPM reference specification also includes the feature of Agents, a feature that we have left out in this work. Given that some proposals are beginning to emerge for agents [Myers 2010; Moreau et al. 2013], a topic for further

[21] http://twiki.ipaw.info/bin/view/OPM/ChangeProposalMoveTimeToProfile

research is to provide a temporal interpretation that can accommodate the notion of agent, and its relation with processes.

Second, adopting Lamport's definition of parallel events [Lamport 1978], two variables u, v in $Vars(G)$ could be defined as parallel events if neither $u \preceq v$ nor $v \preceq u$ are logical consequences of G's theory. In future work, one could take a parallel perspective on OPM, and investigate patterns of parallelism in OPM graphs.

Finally, in Section 6 we proposed a definition of refinements in OPM. As already mentioned, it would be useful to explore the existence of a finite set of graph transformations that could be used to derive all possible refinements. Also, the notion of refinement could take further constraints into consideration, such as process nesting (as considered by Moreau [Moreau 2010a]).

Acknowledgements

Luc Moreau's work is funded in part by the EPSRC SOCIAM (EP/J017728/1) and ORCHID projects (EP/I011587/1), the FP7 SmartSociety project (600854), and the ESRC estat2 project (ES/K007246/1).

REFERENCES

Peter Buneman, James Cheney, Wang-Chiew Tan, and Stijn Vansummeren. 2008. Curated databases. In *PODS '08: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, New York, NY, USA, 1–12. DOI:http://dx.doi.org/10.1145/1376916.1376918

James Cheney. 2010. Causality and the semantics of provenance. In *Proceedings 6th Workshop on Developments in Computational Models (EPTCS)*, S.B. Cooper, E. Kashefi, and P. Panangaden (Eds.), Vol. 26. 63–74.

James Cheney. 2013. *Semantics of the PROV Data Model*. W3C Working Draft WD-prov-sem-20130312. World Wide Web Consortium.

James Cheney, Laura Chiticarius, and Wang-Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1, 4 (2009), 379–474.

James Cheney, Paolo Missier, Luc Moreau (eds.), and Tom De Nies. 2013. *Constraints of the PROV Data Model*. W3C Recommendation REC-prov-constraints-20130430. World Wide Web Consortium. http://www.w3.org/TR/2013/REC-prov-constraints-20130430/

Saumen Dey, Sean Riddle, and Bertram Ludäscher. 2013. Provenance Analyzer: Exploring Provenance Semantics with Logic Rules. In *Presented as part of the 5th USENIX Workshop on the Theory and Practice of Provenance*. USENIX, Berkeley, CA. https://www.usenix.org/conference/tapp13/provenance-analyzer

Li Ding, James Michaelis, Jim McCusker, and Deborah L. McGuinness. 2011. Linked provenance data: A semantic Web-based approach to interoperable workflow traces. *Future Generation Computer Systems* 27, 6 (2011), 797–805.

Andre Freitas, Sean O'Riain, Edward Curry, and Tomas Knap. 2011. W3P: Building an OPM based provenance model for the Web. *Future Generation Computer Systems* 27, 6 (2011), 766–774.

Yolanda Gil, James Cheney, Paul Groth, Olaf Hartig, Simon Miles, Luc Moreau, Paulo Pinheiro da Silva (editors), Sam Coppens, Daniel Garijo, Jose Manuel Gomez, Paolo Missier, Satya Sahoo, and Jun Zhao. 2010. *Provenance XG Final Report*. W3C Incubator Group Report XGR-prov-20101214. World Wide Web Consortium. http://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/

Paul Groth and Luc Moreau. 2011. Representing Distributed Systems Using OPM. *Future Generation Computer Systems* 26, 6 (2011), 757–765.

Joseph Y. Halpern and Judea Pearl. 2005. Causes and Explanations: A Structural-Model Approach. Part I: Causes. *Br J Philos Sci* 56, 4 (2005), 843–887.

Richard Hull and Masatoshi Yoshikawa. 1990. ILOG: Declarative Creation and Manipulation of Object Identifiers. In *Proceedings of the 16th International Conference on Very Large Data Bases*, D. McLeod, R. Sacks-Davis, and H. Schek (Eds.). Morgan Kaufmann, 455–468.

Ian Jacobs and Norman Walsh. 2004. Architecture of the World Wide Web, Volume One. W3C Recommendation 15 December 2004. (2004). http://www.w3.org/TR/webarch/

N. Kwasnikowska, L. Moreau, and J. Van den Bussche. 2010. *A formal account of the Open Provenance Model*. eprint 271819. University of Southampton.

Natalia Kwasnikowska and Jan Van den Bussche. 2008. Mapping the NRC Dataflow Model to the Open Provenance Model. In *Second International Provenance and Annotation Workshop, IPAW'2008 (Lecture Notes in Computer Science)*, Juliana Freire, David Koop, and Luc Moreau (Eds.), Vol. 5272. Springer, 3–16.

Leslie Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM* 21, 7 (1978), 558–565.

Chunhyeok Lim, Shiyong Lu, Artem Chebotko, and Farshad Fotouhi. 2011. Storing, Reasoning, and Querying OPM-Compliant Scientific Workflow Provenance Using Relational Databases. *Future Generation Computer Systems* 27, 6 (2011), 781–789.

Friedemann Mattern. 1989. Virtual time and global states of distributed systems. In *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, M. Cosnard et al. (Eds.). Elsevier Science Publishers, Amsterdam, 215–226.

Robert E. McGrath and Joe Futrelle. 2008. Reasoning about provenance with OWL and SWRL rules. In *AAAI Spring Symposium: AI Meets Business Rules and Process Management*. 87–92.

Simon Miles. 2011. Mapping Attribution Metadata to the Open Provenance Model. *Future Generation Computer Systems* 27, 6 (2011), 806–811.

Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. 2007. The requirements of using provenance in e-Science experiments. *Journal of Grid Computing* 5, 1 (2007), 1–25.

Paolo Missier and others. 2010. Linking multiple workflow provenance traces for interoperable collaborative science. In *Proceedings 5th Workshop on Workflows in Support of Large-Scale Science*. IEEE, 1–8.

Paolo Missier and Carole Goble. 2011. Workflows to Open Provenance Graphs, round-trip. *Future Generation Computer Systems* 27, 6 (2011), 812–819.

Luc Moreau. 2010a. Provenance-Based Reproducibility in the Semantic Web. *Journal of Web Semantics* 9, 2 (2010), 202–221.

Luc Moreau. 2010b. The Foundations for Provenance on the Web. *Foundations and Trends in Web Science* 2, 2–3 (Nov. 2010), 99–241.

Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. 2011. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems* 27, 6 (2011), 743–756.

Luc Moreau, Li Ding, Joe Futrelle, Daniel Garijo Verdejo, Paul Groth, Mike Jewell, Simon Miles, Paolo Missier, Jeff Pan, and Jun Zhao. 2010. Open Provenance Model (OPM) OWL Specification. (2010). http://openprovenance.org/model/opmo

Luc Moreau, Trung Dong Huynh, and Danius Michaelides. 2014. An Online Validator for Provenance: Algorithmic Design, Testing, and API. In *17th International Conference on Fundamental Approaches to Software Engineering (FASÉ14) (Lecture Notes in Computer Science)*. Springer-Verlag. http://eprints.soton.ac.uk/340068/

Luc Moreau, Paolo Missier (eds.), Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. 2013. *PROV-DM: The PROV Data Model*. W3C Recommendation REC-prov-dm-20130430. World Wide Web Consortium. http://www.w3.org/TR/2013/REC-prov-dm-20130430/

James Myers. 2010. I Think Therefore I Am Someone Else: Understanding the confusion of granularity with Continuant/Occurrent and Related Perspective Shifts. In *Provenance and Annotation of Data and Processes*. Lecture Notes in Computer Science, Vol. 6378. Springer, 292–294.

Satya Sahoo, Paul Groth, Olaf Hartig, Simon Miles, Sam Coppens, James Myers, Yolanda Gil, Luc Moreau, Jun Zhao, Michael Panzer, and Daniel Garijo. 2010. *Provenance Vocabulary Mappings*. Technical Report. W3C. http://www.w3.org/2005/Incubator/prov/wiki/Provenance_Vocabulary_Mappings

Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. 2005. A survey of data provenance in e-Science. *SIGMOD Record* 34, 3 (2005), 31–36. DOI:http://dx.doi.org/10.1145/1084805.1084812

A. Tarski. 1986. What are logical notions? *History and Philosophy of Logic* 7 (1986), 143–154. Edited by J. Corcoran.

Gerard Tel. 1994. *Introduction to Distributed Algorithms*. Cambridge University Press.

Curt Tilmes and others. 2013. Provenance Representation for the National Climate Assessment in the Global Change Information System. *IEEE Transactions Geoscience and Remote Sensing* 51, 1 (2013), 5160–5168.

Jeffrey D. Ullman. 1989. *Principles of Database and Knowledge-Base Systems*. Vol. II. Computer Science Press.

W3C PROV 2011. W3C Provenance Working Group Activity. (2011). http://www.w3.org/2011/prov/

W3C Provenance Incubator Activity 2010. Provenance Incubator Group Charter. http://www.w3.org/2005/Incubator/prov/charter. (2010).

Jim Woodcock and Jim Davies. 1996. *Using Z. Specification, Refinement, and Proof*. Prentice Hall.

A. PROOF OF THEOREM 4.7

In this section we present the proof of Theorem 4.7. First, we tackle the soundness property; then, we address the completeness property.

A.1. Proof of soundness

Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G that satisfies the conditions from Theorem 4.7. We have to show that $\mathrm{Th}(G) \models \varphi$. Thereto, let τ be a temporal model of G, i.e., $\tau \models \mathrm{Th}(G)$. We have to show that τ satisfies φ. We inspect the ten possibilities for φ:

(0) if $\varphi \in \mathrm{Th}(G)$, then τ satisfies φ since $\tau \models \mathrm{Th}(G)$.

(1) φ is $\mathrm{create}(B) \preceq \mathrm{create}(A)$ with $G \vdash A \dashrightarrow B$.
As a consequence of Definition 4.5, $G \vdash A \dashrightarrow B$ holds if (A, B) belongs to the transitive closure of *DerivedEdges*. Therefore, there is a path $A_1, A_2, \ldots, A_n$ of derived-from edges from A to B, for some $n \geq 2$ with $A_1 = A$ and $A_n = B$, and with $(A_i, A_{i+1}) \in$ *DerivedEdges*, for $i \in \{1, \ldots, n-1\}$. Since every (A_i, A_{i+1}) is an edge in G, we know that $\mathrm{create}(A_{i+1}) \preceq \mathrm{create}(A_i)$ belongs to $\mathrm{Th}(G)$ (Axiom 4 and Lemma 4.2) and is thus satisfied by τ, i.e., $\tau(\mathrm{create}(A_{i+1})) \leq \tau(\mathrm{create}(A_i))$. Hence we also have $\tau(\mathrm{create}(A_n)) \leq \tau(\mathrm{create}(A_1))$, because $\leq$ is a partial order for τ. Thus τ satisfies $\mathrm{create}(B) \preceq \mathrm{create}(A)$.

(2) φ is $\mathrm{begin}(P) \preceq \mathrm{create}(A)$ with $G \vdash A \dashrightarrow P$.
By Definition 4.5, $G \vdash A \dashrightarrow P$ if either
   a) there is already an edge $A \to P$ or $A \overset{!}{\to} P$ in G; or
   b) there is an artifact B such that $G \vdash A \dashrightarrow B$ and there is an edge $B \to P$ or $B \overset{!}{\to} P$ in G.

   2a) For an edge $A \to P$ ($A \overset{!}{\to} P$) in G, we know by Axiom 5 (Axiom 2), that $\varphi \in \mathrm{Th}(G)$ and thus τ satisfies φ.

   2b) We already know from case 1 that τ satisfies $\mathrm{create}(B) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow B$, i.e., we have $\tau(\mathrm{create}(B)) \leq \tau(\mathrm{create}(A))$. For an edge $B \to P$ ($B \overset{!}{\to} P$) in G, we know by Axiom 5 (Axiom 2), that $\mathrm{begin}(P) \preceq \mathrm{create}(B)$ belongs to $\mathrm{Th}(G)$. Therefore τ satisfies $\mathrm{begin}(P) \preceq \mathrm{create}(B)$, i.e., $\tau(\mathrm{begin}(P)) \leq \tau(\mathrm{create}(B))$. Hence $\tau(\mathrm{begin}(P)) \leq \tau(\mathrm{create}(A))$, since $\leq$ is a partial order for τ. We conclude that τ satisfies $\mathrm{begin}(P) \preceq \mathrm{create}(A)$.

(3) φ is $\mathrm{create}(A) \preceq \mathrm{end}(P)$ with $G \vdash P \dashrightarrow A$.
By Definition 4.5, $G \vdash P \dashrightarrow A$ if either
   a) there is already an edge $P \to A$ or $P \overset{!}{\to} A$ in G; or
   b) there is an artifact B such that $G \vdash B \dashrightarrow A$ and there is an edge $P \to B$ or $P \overset{!}{\to} B$ or $B \overset{!}{\to} P$ in G.

   3a) For an edge $P \to A$ ($P \overset{!}{\to} A$) in G, we know by Axiom 6 (Axiom 3) that $\varphi \in \mathrm{Th}(G)$ and thus τ satisfies φ.

   3b) We already know from case 1 that τ satisfies $\mathrm{create}(A) \preceq \mathrm{create}(B)$ for $G \vdash B \dashrightarrow A$. For an edge $P \to B$ ($P \overset{!}{\to} B$) in G, we know by Axiom 6 (Axiom 3) that $\mathrm{create}(B) \preceq \mathrm{end}(P)$ belongs to $\mathrm{Th}(G)$. For an edge $B \overset{!}{\to} P$ in G, we know by Axiom 2 that $\mathrm{create}(B) \preceq \mathrm{end}(P)$ belongs to $\mathrm{Th}(G)$. Therefore, in each case, τ satisfies both $\mathrm{create}(A) \preceq \mathrm{create}(B)$ and $\mathrm{create}(B) \preceq \mathrm{end}(P)$. Hence τ also satisfies $\mathrm{create}(A) \preceq \mathrm{end}(P)$.

(4) φ is $\mathrm{begin}(Q) \preceq \mathrm{end}(P)$ with $G \vdash P \dashrightarrow Q$.

By Definition 4.5, $G \vdash P \dashrightarrow Q$ if either

a) there is already an edge $P \to Q$ in G; or
b) there is an artifact A such that $G \vdash A \dashrightarrow Q$, and either $G \vdash P \dashrightarrow A$ or there is an edge $A \xrightarrow{!} P$ in G.

4a) For an edge $P \to Q$ in G, we know by Axiom 7 that $\varphi \in \mathrm{Th}(G)$ and thus τ satisfies φ.

4b) We already know from case 2 that τ satisfies $\mathrm{begin}(Q) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow Q$. We also know from case (3) that τ satisfies $\mathrm{create}(A) \preceq \mathrm{end}(P)$ for $G \vdash P \dashrightarrow A$. For an edge $A \xrightarrow{!} P$ in G, we know by Axiom 2 that $\mathrm{create}(A) \preceq \mathrm{end}(P)$ belongs to $\mathrm{Th}(G)$. Therefore, in each case, τ satisfies both $\mathrm{begin}(Q) \preceq \mathrm{create}(A)$ and $\mathrm{create}(A) \preceq \mathrm{end}(P)$. Thus τ also satisfies $\mathrm{begin}(Q) \preceq \mathrm{end}(P)$.

(5) φ is $\mathrm{create}(B) \preceq \mathrm{use}(P, r, A)$ with $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow B$.
We already know from case 1 that τ satisfies $\mathrm{create}(B) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow B$. For edge $P \xrightarrow{r} A$ in G we know, by Axiom 3, that $\mathrm{create}(A) \preceq \mathrm{use}(P, r, A)$ belongs to $\mathrm{Th}(G)$, and is thus satisfied by τ. Therefore, τ also satisfies $\mathrm{create}(B) \preceq \mathrm{use}(P, r, A)$.

(6) φ is $\mathrm{begin}(Q) \preceq \mathrm{use}(P, r, A)$ with $P \xrightarrow{r} A$ in G and $G \vdash A \dashrightarrow Q$.
We already know from case 2 that τ satisfies $\mathrm{begin}(Q) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow Q$. For edge $P \xrightarrow{r} A$ in G, we know, by Axiom 3, that $\mathrm{create}(A) \preceq \mathrm{use}(P, r, A)$ belongs to $\mathrm{Th}(G)$, and is thus satisfied by τ. Thus, τ also satisfies $\mathrm{begin}(Q) \preceq \mathrm{use}(P, r, A)$.

(7) φ is $\mathrm{use}(P, r, C) \preceq \mathrm{create}(A)$ with $G \triangle (B, C, P, r)$ and $G \vdash A \dashrightarrow B$.
We already know from case 1 that τ satisfies $\mathrm{create}(B) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow B$. From $G \triangle (B, C, P, r)$ we know, by Axiom 8, that $\mathrm{use}(P, r, C) \preceq \mathrm{create}(B)$ belongs to $\mathrm{Th}(G)$, and is thus satisfied by τ. Therefore, τ also satisfies $\mathrm{use}(P, r, C) \preceq \mathrm{create}(A)$.

(8) φ is $\mathrm{use}(P, r, B) \preceq \mathrm{end}(Q)$ with $G \triangle (A, B, P, r)$ and $G \vdash Q \dashrightarrow A$.
We already know from case 3 that τ satisfies $\mathrm{create}(A) \preceq \mathrm{end}(Q)$ for $G \vdash Q \dashrightarrow A$. From $G \triangle (A, B, P, r)$ we know, by Axiom 8, that $\mathrm{use}(P, r, B) \preceq \mathrm{create}(A)$ belongs to $\mathrm{Th}(G)$, and is thus satisfied by τ. Hence, τ also satisfies $\mathrm{use}(P, r, B) \preceq \mathrm{end}(Q)$.

(9) φ is $\mathrm{use}(P, r, B) \preceq \mathrm{use}(Q, s, A)$ with $G \triangle (C, B, P, r)$ in G, $Q \xrightarrow{s} A$ in G, and either (a) $A = C$ or (b) $G \vdash A \dashrightarrow C$.
We already know from case 1 that τ satisfies $\mathrm{create}(C) \preceq \mathrm{create}(A)$ for $G \vdash A \dashrightarrow C$ (9b). If $A = C$ (9a) then, obviously, $\tau(A) = \tau(C)$, and τ still satisfies $\mathrm{create}(C) \preceq \mathrm{create}(A)$. For edge $Q \xrightarrow{s} A$ in G, we know, by Axiom 3, that $\mathrm{create}(A) \preceq \mathrm{use}(Q, s, A)$ belongs to $\mathrm{Th}(G)$, and is thus satisfied by τ. From $G \triangle (C, B, P, r)$ we know, by Axiom 8S, that $\mathrm{use}(P, r, B) \preceq \mathrm{create}(C)$ belongs to $\mathrm{Th}(G)$, hence is satisfied by τ. Therefore, we have $\mathrm{use}(P, r, B) \preceq \mathrm{create}(C) \preceq \mathrm{create}(A) \preceq \mathrm{use}(Q, s, A)$. We conclude that τ also satisfies $\mathrm{use}(P, r, B) \preceq \mathrm{use}(Q, s, A)$.

A.2. Proof of completeness

Let G be a legal OPM graph and let φ be a nontrivial inequality over the temporal variables of G such that $\mathrm{Th}(G) \models \varphi$. We must show that $\varphi \in \mathrm{Th}(G)$ or that φ matches one of the cases 1–9 of Theorem 4.7.

It is well known [Ullman 1989] that φ can be inferred from $\mathrm{Th}(G)$ by using repeated applications of the rule of transitivity: "from $u \preceq v$ and $v \preceq w$ infer $u \preceq w$." We proceed by induction on the number of applications of the transitivity rule.

If φ can be inferred by zero applications, then φ is already in $\mathrm{Th}(G)$ and we are done, as this corresponds to case 0 of the theorem.

Now consider an application of transitivity inferring φ of the form $u \preceq w$ from $u \preceq v \preceq w$, where, by induction, the theorem can already be assumed to hold for the inequalities $u \preceq v$ and $v \preceq w$. Since begin-variables (end-variables) never appear on the right-hand (left-hand)

side of an inequality, v cannot be a begin-variable (end-variable). That leaves us with two cases, with v being either a create- or a use-variable.

v is a create-variable. Let v be a create-variable, say create(A_v). Let us list the possibilities for u and note the relevant properties:

(a) u is also a create-variable, say create(A_u). By induction, we know that the inequality $u \preceq v$ either already belongs to Th(G), so there is an edge $A_v \to A_u$ in G (Axiom 4), or the inequality corresponds to case 1 of the theorem, therefore $G \vdash A_v \dashrightarrow A_u$. In either case we have $G \vdash A_v \dashrightarrow A_u$.
(b) u is a begin-variable, say begin(P_u). By induction, $u \preceq v$ either belongs to Th(G), so there is an edge $A_v \xrightarrow{!} P_u$ in G (Axiom 2) or $A_v \to P_u$ in G (Axiom 5); or $u \preceq v$ corresponds to case 2 of the theorem, therefore $G \vdash A_v \dashrightarrow P_u$. In either case we have $G \vdash A_v \dashrightarrow P_u$.
(c) u is a use-variable, say use(P_u, r_u, A_u). By induction, $u \preceq v$ either (c1) belongs to Th(G), so there is some use–generate–derive triangle (A_v, A_u, P_u, r_u) in G (Axiom 8); or (c2) $u \preceq v$ corresponds to case 7 of the theorem, thus there is a use–generate–derive triangle (A'_v, A_u, P_u, r_u) in G with $G \vdash A_v \dashrightarrow A'_v$.

We also list the possibilities for w and their relevant properties:

(d) w is also a create-variable, say create(A_w). By the induction hypothesis applied to $v \preceq w$, reasoning similarly as in case (a) above, we have $G \vdash A_w \dashrightarrow A_v$.
(e) w is an end-variable, say end(P_w). By induction, $v \preceq w$ either belongs to Th(G), so there is an edge $A_v \xrightarrow{!} P_w$ in G (Axiom 2) or $P_w \to A_v$ in G (Axiom 6); or $v \preceq w$ corresponds to case 3 of the theorem, therefore $G \vdash P_w \dashrightarrow A_v$. We have thus either (e1) $A_v \xrightarrow{!} P_w$ in G or (e2) $G \vdash P_w \dashrightarrow A_v$.
(f) w is a use-variable, say use(P_w, r_w, A_w). This necessitates the presence of edge $P_w \xrightarrow{r_w} A_w$ in G. By induction, the inequality $v \preceq w$ either (f1) belongs to Th(G), so that $A_w = A_v$ (Axiom 3); or (f2) $v \preceq w$ corresponds to case 5 of the theorem, thus $G \vdash A_w \dashrightarrow A_v$.

We can now inspect the nine possible combinations:

(ad) φ is create$(A_u) \preceq$ create(A_w). From $G \vdash A_v \dashrightarrow A_u$ and $G \vdash A_w \dashrightarrow A_v$ we infer $G \vdash A_w \dashrightarrow A_u$, which matches case 1 of the theorem.
(ae) φ is create$(A_u) \preceq$ end(P_w). From $G \vdash A_v \dashrightarrow A_u$ and either $A_v \xrightarrow{!} P_w$ in G or $G \vdash P_w \dashrightarrow A_v$ we infer $G \vdash P_w \dashrightarrow A_u$, which matches case 3 of the theorem.
(af) φ is create$(A_u) \preceq$ use(P_w, r_w, A_w) with $P_w \xrightarrow{r_w} A_w$ in G. In case f1, we have $G \vdash A_v \dashrightarrow A_u$ and $A_v = A_w$, so the case corresponds to case 5 of the theorem. In case f2, we infer $G \vdash A_w \dashrightarrow A_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, which again matches case 5 of the theorem.
(bd) φ is begin$(P_u) \preceq$ create(A_w). From $G \vdash A_v \dashrightarrow P_u$ and $G \vdash A_w \dashrightarrow A_v$ we infer $G \vdash A_w \dashrightarrow P_u$, which corresponds to case 2 of the theorem.
(be) φ is begin$(P_u) \preceq$ end(P_w). From $G \vdash A_v \dashrightarrow P_u$ and either $A_v \xrightarrow{!} P_w$ in G or $G \vdash P_w \dashrightarrow A_v$ we infer $G \vdash P_w \dashrightarrow P_u$, which matches case 4 of the theorem.
(bf) φ is begin$(P_u) \preceq$ use(P_w, r_w, A_w) with $P_w \xrightarrow{r_w} A_w$ in G. In case f1, we have $G \vdash A_v \dashrightarrow P_u$ and $A_v = A_w$, so the case corresponds to case 6 of the theorem. In case f2, we infer $G \vdash A_w \dashrightarrow P_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, which again matches case 6 of the theorem.
(cd) φ is use$(P_u, r_u, A_u) \preceq$ create(A_w). Case c1 corresponds directly to case 7 of the theorem. In case c2, we infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_v \dashrightarrow A'_v$ and $G \vdash A_w \dashrightarrow A_v$, which again matches case 7 of the theorem.

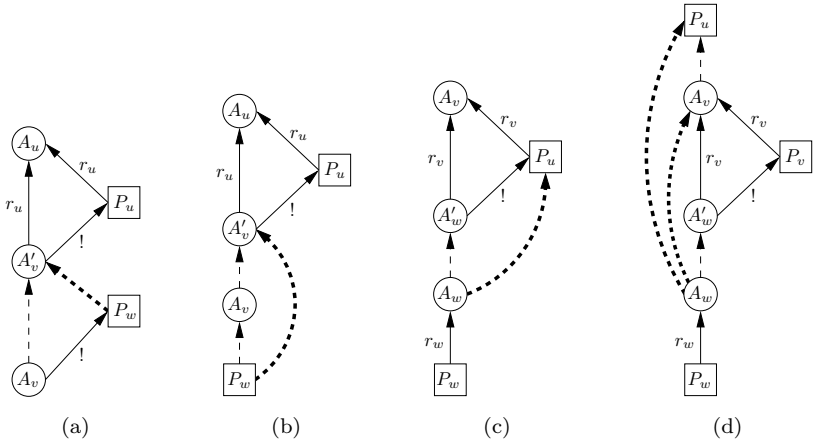

Fig. 16. Proof of the completeness of Theorem 4.7 for cases (a) c2 with e1, (b) c2 with e2, (c) h1 with l, and (d) h2 with l. The bold edges are newly inferred.

(ce) φ is $\mathrm{use}(P_u, r_u, A_u) \preceq \mathrm{end}(P_w)$. First, consider case c1 together with e1. Since $G \triangle (A_v, A_u, P_u, r_u)$, P_u and P_w must be equal because G is legal. In this case the inequality holds by Axiom 3. Case c1 together with e2 corresponds directly to case 8 of the theorem. Finally, in case c2, from $G \vdash A_v \dashrightarrow A'_v$, and either $A_v \xrightarrow{!} P_w$ (from e1, see Figure 16(a)) or $G \vdash P_w \dashrightarrow A_v$ (from e2, see Figure 16(b)) we infer $G \vdash P_w \dashrightarrow A'_v$, which matches case 8 of the theorem.

(cf) φ is $\mathrm{use}(P_u, r_u, A_u) \preceq \mathrm{use}(P_w, r_w, A_w)$. Case c1 together with f1 corresponds directly to case 9a of the theorem. Case c1 together with f2 matches case 9b of the theorem. The same holds for c2 together with f1. In case c2 together with f2 we infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A'_v$, which again matches case 9b of the theorem.

v is a use-variable. Let v be a use-variable, say $\mathrm{use}(P_v, r_v, A_v)$. Note that this necessitates the presence of the edge $P_v \xrightarrow{r_v} A_v$ in G. Let us list the possibilities for u and note the relevant properties:

(g) u is a create-variable, say $\mathrm{create}(A_u)$. By induction, we know that the inequality $u \preceq v$ either (g1) already belongs to $\mathrm{Th}(G)$, so A_u equals A_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (g2) the inequality corresponds to case 5 of the theorem, therefore $G \vdash A_v \dashrightarrow A_u$ with the edge $P_v \xrightarrow{r_v} A_v$ in G.

(h) u is a begin-variable, say $\mathrm{begin}(P_u)$. By induction, $u \preceq v$ either (h1) already belongs to $\mathrm{Th}(G)$, thus P_u equals P_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (h2) the inequality corresponds to case 6 of the theorem, therefore $G \vdash A_v \dashrightarrow P_u$ with the edge $P_v \xrightarrow{r_v} A_v$ in G.

(i) u is also a use-variable, say $\mathrm{use}(P_u, r_u, A_u)$. By induction, we know that the inequality $u \preceq v$ can only correspond to case 9 of the theorem, therefore we have some triangle (A'_v, A_u, P_u, r_u) in G with the edge $P_v \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$.

We also list the possibilities for w and their relevant properties:

(j) w is a create-variable, say $\mathrm{create}(A_w)$. By induction, $v \preceq w$ either (j1) already belongs to $\mathrm{Th}(G)$, so there is some use–generate–derive triangle (A_w, A_v, P_v, r_v) in G (Axiom 8); or (j2) the inequality corresponds to case 7 of the theorem, and there is a use–generate–derive triangle (A'_w, A_v, P_v, r_v) in G with $G \vdash A_w \dashrightarrow A'_w$. Note that in both cases we

can infer $G \vdash A_w \dashrightarrow A_v$. Indeed, in case j1 we have the edge $A_w \xrightarrow{r_v} A_v$ in G. In case j2 we have $G \vdash A_w \dashrightarrow A'_w$ and the edge $A'_w \xrightarrow{r_v} A_v$ in G.

(k) w is an end-variable, say $\mathrm{end}(P_w)$. By induction, $v \preceq w$ either (k1) already belongs to $\mathrm{Th}(G)$, so P_w equals P_v with the edge $P_v \xrightarrow{r_v} A_v$ in G (Axiom 3); or (k2) corresponds to case 8 of the theorem, thus there is some use–generate–derive triangle (A_w, A_v, P_v, r_v) in G with $G \vdash P_w \dashrightarrow A_w$. Note that in both cases we can infer $G \vdash P_w \dashrightarrow A_v$. Indeed, in case k1 we have the edge $P_w \xrightarrow{r_v} A_v$ in G. In case k2 we have $G \vdash P_w \dashrightarrow A_w$ and the edge $A_w \xrightarrow{r_v} A_v$ in G.

(l) w is also a use-variable, say $\mathrm{use}(P_w, r_w, A_w)$. By induction, we know that the inequality $v \preceq w$ can only correspond to case 9 of the theorem, so there is some use–generate–derive triangle (A'_w, A_v, P_v, r_v) in G with $P_w \xrightarrow{r_w} A_w$ in G, and either $A_w = A'_w$ or $G \vdash A_w \dashrightarrow A'_w$. Note that in both cases we can infer $G \vdash A_w \dashrightarrow A_v$ by the edge $A'_w \xrightarrow{r_v} A_v$ in the triangle.

We can now inspect the nine possible combinations:

(gj) φ is $\mathrm{create}(A_u) \preceq \mathrm{create}(A_w)$. In case g1, we have $G \vdash A_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 1 of the theorem. In case g2, we have $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, so we can infer $G \vdash A_w \dashrightarrow A_u$, which, again, matches case 1 of the theorem.

(gk) φ is $\mathrm{create}(A_u) \preceq \mathrm{end}(P_w)$. In case g1 together with k1, we have $A_v = A_u$, $P_v = P_w$, and the edge $P_v \xrightarrow{r_v} A_v$ in G, so the case corresponds directly to case 3 of the theorem. In case g2 together with k1, we have $P_v = P_w$ with the edge $P_v \xrightarrow{r_v} A_v$ in G. From the latter and $G \vdash A_v \dashrightarrow A_u$, we infer $G \vdash P_w \dashrightarrow A_u$, which again matches case 3 of the theorem. In case g1 together with k2, we have $G \vdash P_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 3 of the theorem. In case g2 together with k2, we infer $G \vdash P_w \dashrightarrow A_u$ from $G \vdash P_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$. Hence the case again matches case 3 of the theorem.

(gl) φ is $\mathrm{create}(A_u) \preceq \mathrm{use}(P_w, r_w, A_w)$. In case g1, we have $G \vdash A_w \dashrightarrow A_v$ and $A_v = A_u$, so the case corresponds directly to case 5 of the theorem. In case g2, we infer $G \vdash A_w \dashrightarrow A_u$ from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow A_u$, which matches case 5 of the theorem.

(hj) φ is $\mathrm{begin}(P_u) \preceq \mathrm{create}(A_w)$. By case j, we infer $G \vdash A_w \dashrightarrow P_v$. (Indeed, in case j1 we easily infer $G \vdash A_w \dashrightarrow P_v$. In case j2 we also infer $G \vdash A_w \dashrightarrow P_v$ from $G \vdash A_w \dashrightarrow A'_w$ and $A'_w \xrightarrow{!} P_v$ in G.) Now in case h1 we have $P_v = P_u$, so the case corresponds directly to case 2 of the theorem. In case h2 we have $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, so we can infer $G \vdash A_w \dashrightarrow P_u$. Thus the case again matches case 2 of the theorem.

(hk) φ is $\mathrm{begin}(P_u) \preceq \mathrm{end}(P_w)$. In case h1 together with k1, we have $P_u = P_v = P_w$, so the inequality trivially holds (Axiom 1). In case h1 together with k2, we have $P_v = P_u$, and we infer $G \vdash P_w \dashrightarrow P_v$ from $G \vdash P_w \dashrightarrow A_w$ and $G \vdash A_w \dashrightarrow P_v$. Thus the case corresponds to case 4 of the theorem. In case h2 we have $G \vdash P_w \dashrightarrow A_v$, which combined with $G \vdash A_v \dashrightarrow P_u$, yields $G \vdash P_w \dashrightarrow P_u$. Hence the case again matches case 4 of the theorem.

(hl) φ is $\mathrm{begin}(P_u) \preceq \mathrm{use}(P_w, r_w, A_w)$. By case l, we infer $G \vdash A_w \dashrightarrow P_v$ from $A'_w \xrightarrow{!} P_v$ in G, and either $A_w = A'_w$ or $G \vdash A_w \dashrightarrow A'_w$. Also, there is an edge $P_w \xrightarrow{r_w} A_w$ in G, and $G \vdash A_w \dashrightarrow A_v$. In case h1 (see Figure 16(c)) we have $P_v = P_u$, so the case corresponds to case 6 of the theorem. In case h2 (see Figure 16(d)), from $G \vdash A_w \dashrightarrow A_v$ and $G \vdash A_v \dashrightarrow P_u$, we infer $G \vdash A_w \dashrightarrow P_u$. Hence the case again matches case 6 of the theorem.

(ij) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{create}(A_w)$. We infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_w \dashrightarrow A_v$, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Together with $G \triangle (A'_v, A_u, P_u, r_u)$, the case corresponds to case 7 of the theorem.

(ik) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{end}(P_w)$. We already have $G \triangle (A'_v, A_u, P_u, r_u)$. In case k1, we have $P_w = P_v$. We infer $G \vdash P_v \dashrightarrow A'_v$ from the edge $P_v \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. The case thus matches case 8 of the theorem. In case k2, we infer $G \vdash P_w \dashrightarrow A'_v$ from $G \vdash P_w \dashrightarrow A_w$, $A_w \xrightarrow{r_v} A_v$ in G, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Hence the case again matches case 8 of the theorem.

(il) φ is $\text{use}(P_u, r_u, A_u) \preceq \text{use}(P_w, r_w, A_w)$. We already have $P_w \xrightarrow{r_w} A_w$ in G and $G \triangle (A'_v, A_u, P_u, r_u)$. We additionally infer $G \vdash A_w \dashrightarrow A'_v$ from $G \vdash A_w \dashrightarrow A_v$, and either $A_v = A'_v$ or $G \vdash A_v \dashrightarrow A'_v$. Hence the case matches case 9b of the theorem.